remote sensing

MDPI

Article

AERO: AI-Enabled Remote Sensing Observation with Onboard Edge Computing in UAVs

Anis Koubaa *,†, Adel Ammar †, Mohamed Abdelkader, Yasser Alhabashi and Lahouari Ghouti

College of Computer & Information Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia; aammar@psu.edu.sa (A.A.)

* Correspondence: akoubaa@psu.edu.sa

† These authors contributed equally to this work.

Abstract: Unmanned aerial vehicles (UAVs) equipped with computer vision capabilities have been widely utilized in several remote sensing applications, such as precision agriculture, environmental monitoring, and surveillance. However, the commercial usage of these UAVs in such applications is mostly performed manually, with humans being responsible for data observation or offline processing after data collection due to the lack of on board AI on edge. Other technical methods rely on the cloud computation offloading of AI applications, where inference is conducted on video streams, which can be unscalable and infeasible due to remote cloud servers' limited connectivity and high latency. To overcome these issues, this paper presents a new approach to using edge computing in drones to enable the processing of extensive AI tasks onboard UAVs for remote sensing. We propose a cloud–edge hybrid system architecture where the edge is responsible for processing AI tasks and the cloud is responsible for data storage, manipulation, and visualization. We designed AERO, a UAV brain system with onboard AI capability using GPU-enabled edge devices. AERO is a novel multi-stage deep learning module that combines object detection (YOLOv4 and YOLOv7) and tracking (DeepSort) with TensorRT accelerators to capture objects of interest with high accuracy and transmit data to the cloud in real time without redundancy. AERO processes the detected objects over multiple consecutive frames to maximize detection accuracy. The experiments show a reduced false positive rate (0.7%), a low percentage of tracking identity switches (1.6%), and an average inference speed of 15.5 FPS on a Jetson Xavier AGX edge device.

Keywords: unmanned aerial vehicles; object detection; object tracking; remote sensing; object localization; edge computing; inspection; YOLOv4; YOLOv7; DeepSORT

Citation: Koubaa, A.; Ammar, A.; Abdelkader, M.; Alhabashi, Y.; Ghouti, L. AERO: AI-Enabled Remote Sensing Observation with Onboard Edge Computing in UAVs. *Remote Sens.* **2023**, *15*, 1873. https://doi.org/10.3390/rs15071873

Academic Editors: Wenjiang Huang, Giovanni Laneve, Yingying Dong and Chenghai Yang

Received: 3 March 2023
Revised: 27 March 2023
Accepted: 29 March 2023
Published: 31 March 2023

1. Introduction

The use of unmanned aerial vehicles (UAVs), also known as drones, in remote sensing has been increasingly beneficial as they help to speed up the data collection of assets of interest using aerial images. Drones make the data collection process cost-effective and flexible as drones can fly at low or high altitudes. It also helps missions to be more efficient as large regions can be precisely covered in short times thanks to the use of high-resolution cameras. Data collection also becomes safe as drones replace humans entering dangerous or difficult-to-access environments. These benefits are driving the remote sensing business to increasingly rely on drones. As a matter of fact, the market of commercial use of UAVs, including remote sensing applications, was valued at USD 5.85 billion in 2020 and is expected to have a compound annual growth rate of 14.2% [1].

1.1. Motivating Scenarios

AI-powered drones with onboard intelligence are becoming a game changer for many applications, including search and rescue, rapid infrastructure inspection, and remote sensing, to name a few. Integrating AI solutions directly on the drone's edge (compute)

devices can dramatically reduce the decision-making time and operational costs. In this paper, we will discuss several scenarios in different use cases' contexts, showing the limitation of existing solutions of UAVs for vision-based applications and discussing the advantages of real-time onboard AI.

1.1.1. Remote Sensing

Tree counting is one of the applications in remote sensing, where a drone surveys farm regions to count the number of trees. In [2], the authors proposed an offline counting and geo-localization of palm trees based on aerial images using deep learning. However, the processing was performed offline after collecting palm tree images from a UAV. The process of data collection and its offline processing takes a long time and needs to be performed in real time. Leveraging GPU-based edge devices on board the UAV enables the full automation of palm tree counting in real time. Furthermore, it helps to send each palm tree information (e.g., image and coordinates) to the cloud and store it in databases in real time. Naturally, the same concept can be applied to other remote sensing applications, such as gas leakage localization and mapping [3], flash flood real-time monitoring [4,5], and urban environment segmentation [6,7].

1.1.2. Search and Rescue

Consider a search-and-rescue mission where a drone is required to explore an extended region to search for a missing person in the desert or a forest, for example. It has been reported that more than 100 people get lost and die in the desert annually in Saudi Arabia alone [8]. The current practice for search-and-rescue using UAVs is to manually explore a region with human observers to find the target missing people. Using AI on board will help to automate the process as the UAV can execute specialized person detection models on board and automatically report their location in real time. It is also possible to use a swarm of drones to perform search-and-rescue missions in parallel, speeding up the search process and increasing the probability of finding and saving people [9].

1.1.3. Inspection and Surveillance

Surveillance and inspection using UAVs is one of the fastest businesses in the drone industry [10]. Drones are typically used to detect objects of interest in surveillance missions, such as vehicles [11], pedestrians [12], and buildings [13]. Traditional approaches either inspect real-time video streams by human observers or record scenes' videos and process them offline either manually or using AI techniques to extract target objects. The use of onboard AI processing in the UAVs will help to automate the inspection process and identify target objects in real time with a high accuracy, as will be demonstrated in this paper.

The automation of these applications on board UAVs is possible thanks to the evolution of edge devices and their support of advanced graphics processing units (GPUs), making it possible to process complex deep learning models in real time.

Before the evolution of edge computing, computation offloading has evolved as the prominent approach to processing heavy computation in the cloud instead of processing them on robots or drones. This concept has been known as cloud robotics. While computation offloading offers several advantages by leveraging the capabilities of the cloud resources to speed up the processing of deep learning models and computation-intensive applications, it suffers from high communication overhead. It also needs a large bandwidth and high-quality communication, which cannot always be afforded. In [14], the authors proposed a system architecture for computation offloading in Internet-connected drones and compared the performance of cloud computation offloading versus edge computing for deep learning applications. The study investigated the tradeoff between the communication cost and computation and found that computation offloading provides higher throughput despite larger communication delays.

1.2. Main Contributions

In this paper, we aimed to tackle the persisting challenge of deploying onboard artificial intelligence on the edge in commercial unmanned aerial vehicles (UAVs) that are primarily utilized for remote sensing applications. This predicament often necessitates laborious manual data observation or time-consuming offline processing, as cloud-based approaches are often impractical. There are a few recent works that tested onboard AI on edge in UAVs for detection and tracking, such as [15–17]. Nevertheless, they did not investigate the hybrid system architecture that we implemented in this work, and did not discuss the role of the cloud in their solution. To bridge this gap, we propose using edge computation on board drones to enable advanced observation and surveillance applications, involving object detection, multi-object-tracking, and real-time reporting of detected target objects to the cloud. In brief, the contributions of the paper can be summarized as follows:

- We propose a new approach to using edge computing in drones to enable the processing of extensive AI tasks on board UAVs for remote sensing. To overcome the limited connectivity and high latency of remote cloud servers, we propose a cloud–edge hybrid system architecture. In this architecture, the edge is responsible for processing AI tasks, and the cloud is responsible for data storage, manipulation, and visualization. Our proposed architecture can provide a more scalable and efficient solution for remote sensing applications.
- To implement our proposed architecture, we designed and developed AERO, a UAV brain system with onboard AI capability using GPU-enabled edge devices. AERO allows us to capture objects of interest with high accuracy and transmit data to the cloud in real time without redundancy. AERO processes the detected objects over multiple consecutive frames to maximize detection accuracy. AERO can be a significant advancement in the field of remote sensing as it enables UAVs to perform onboard AI tasks with high accuracy and real-time data transmission, providing a more efficient and cost-effective solution for remote sensing applications.

The remaining sections of the paper are organized as follows. Section 1.3 provides a review of the relevant literature and situates the contribution of the paper in comparison to previous work. Section 2 presents the architecture of the AERO system and describes the AERO AI Module. In Section 3, we detail the experimental study conducted to evaluate the AERO system's performance, and we discuss their results in Section 4. Finally, Section 5 concludes the paper and suggests potential future research directions for further improvements.

1.3. Related Works

The introduction of UAVs in remote sensing has paved the way for several promising applications that span a wide range of domains [18]. Impressive progress has been achieved in academic and industrial arenas. Diversity in available solutions is mainly attributed to the underlying technologies and modalities used in the data sense/acquisition processes [19]. The latter processes are domain-specific in nature [20,21]. Other techniques, including data preprocessing, feature extraction, and classification, are specifically designed for the application, whether civilian or military. UAV applications in remote sensing have been reviewed in [21,22].

1.3.1. Edge Computing and UAVs

Several recent works addressed the edge computing paradigm, which involves moving computational processing and storage closer to the end-users, devices, or sensors rather than relying solely on cloud-based solutions. Specifically, these works focused on leveraging UAVs to offload computation tasks to edge computing servers, which enables low-latency computations of specific tasks without noticeable delay.

In [23], Messous et al. proposed an evaluation mechanism of the integration of the computation offloading to edge computing servers for the efficient deployment of UAVs.

Based on the proposed evaluation, UAV-based models are able to decide whether to perform local processing, offload to an edge server, or delegate the computational tasks to the ground station. Informed decisions are based on low-latency computations of specific tasks without noticeable delay. Qian et al. [24] investigated the performance of a UAV-mounted mobile edge computing network where the UAV unit offloads and executes specific tasks that originate from some mobile terminal users. The trajectory planning problem was formulated as a Markov decision process (MDP) where optimal trajectories were obtained using a policy based on the double deep Q-network (DDQN) algorithm [25]. Thanks to the DDQN efficiency, higher throughput scores were attained.

A machine-learning-based solution for the planning of UAV trajectories is attributed to Afifi, and Gadallah [26]. Unlike many existing solutions, Afifi and Gadallah targeted missions with real-time navigation requirements in dense urban environments, where existing 5G infrastructures are astutely employed to ensure UAV navigation in complex environments through continuous interactions between the UAV units and the selected 5G network. Like [24], the proposed trajectory planning solution relies on deep reinforcement learning strategies, where the planning accuracy attains 99%.

In [27], Xia et al. proposed a flexible design of a wireless edge network using two UAV units. In this design, both units are restricted to operate at fixed altitudes with accelerated motions. Over a defined area, while the first UAV unit is in charge of forwarding downlink signals to the user terminals (UTs), the second unit is assigned to the collection of the uplink data. Using statistical information collected from the UT elements and UAVs, lower bounds on conditional average achievable rates are derived. The proposed scheme is demonstrated to attain an energy efficiency higher than existing ones.

Bin et al. [28] tackled the problem of the variability of user mobility and MEC environments, where they suggested a novel scheme for intelligent task offloading in UAV-enabled MEC systems using a digital twin (DT). At the core of the proposed scheme lies the DDQN model, which is specifically designed to effectively constrain multi-objective problems. The model was jointly optimized using closed-form and iterative procedures. The simulation results clearly indicate the convergence of the DDQN-based model while drastically minimizing the total energy consumption of the MEC system compared to existing optimization techniques.

A new aerial edge Internet of Things (EdgeIoT) system was contributed by Li et al. [29]. In this new EdgeIoT system, a UAV unit is operated as a mobile edge server for processing computational processes related to mission-critical tasks emanating from ground IoT devices. To capture the underlying feature correlations, a graph-based neural network architecture (GNN) was used for the supervised training of the A2C structure. The reported performance analysis highlights the superiority of the mixed GNN-A2C framework in terms of the convergence speed and missing task rates.

In [30], Qian et al. proposed a Monte Carlo tree search (MCTS)-based path planning technique assuming that a single UAV is deployed as a mobile service to provide computation tasks offloading services for a set of mobile users on the ground. The reported results show that the MCTS-based scheme outperforms state-of-the-art DQN-based planning algorithms in terms of the average throughput and convergence speed. In some instances, UAVs assist edge clouds (ECs) for the large-scale sparely distributed user equipment, which allows for wide coverage and reliable wireless communication. However, UAVs have limited computation and energy resources, which opens the floor for potential optimal resource allocation.

In [31], Wang et al. introduced a vehicular fog computing (VFC) system where unmanned ground vehicles (UGVs) perform the computation tasks offloaded from UAVs that are deployed in natural disaster areas. In these areas, UAVs are effectively used to survey disaster areas and even perform emergency missions, given their swift deployment and flexibility. However, this efficiency is hindered by the limited energy and computational capabilities of UAVs. These limitations are properly addressed by the VFC-based UAV system proposed by Wang et al., where UGVs may be assigned to perform the computation

tasks offloaded from UAVs to save energy and computational power. To ensure a smooth and steady UAV–UGV collaboration and interaction, the computation task offloading problem was cast into a two-sided matching problem, where an iterative stable matching algorithm was used. This matching algorithm aims at assigning to each UAV the most suitable UGV among the available ones for offloading while maximizing the usage of both UAVs and UGVs and reducing the average delay.

Yang et al. [32] considered a UAV-enabled MEC platform where multiple mobile ground users move randomly and tasks arrive in a random fashion. To minimize the average weighted energy consumption of all users under constraints expressed in terms of data queue stability and average UAV energy consumption, Yang et al. suggested a multi-stage stochastic optimization scheme where Lyapunov optimization is converted into simpler per-slot deterministic problems vis-a-vis the number of optimizing variables. Based on their formulation, Yang et al. solved the resource allocation and the UAV movement problems using two reduced-complexity methods, either jointly or separately. The two methods not only satisfy the average UAV energy and queue stability constraints, but they also reconcile the length of the queue backlog and the user energy consumption bounds. The reported results show that the proposed joint and two-stage stochastic optimization schemes outperform existing learning-based solutions. Finally, it should be noted that the joint optimization scheme attains a better performance than its two-stage counterpart at the expense of an increased computational complexity. Most of the solutions discussed so far attempt to optimize the UAVs' total (or average) energy consumption and computational power allocation among mobile users using some type of learning-based strategy.

In their proposal, Lyi et al. [33] adopted a different approach to maximize the computation bits of the whole MEC system: the joint optimization of task offloading time allocation, bandwidth allocation, and the UAV trajectory under specific energy constraints of ground devices and maximal UAV battery energy. The proposed solution splits the overall optimization procedure into three stages, where successive convex optimization schemes are used. Once individual solutions are identified, a block coordinate descent (BCD) algorithm integrates the solution of the initial optimization problem. Such a formulation aims at obtaining alternating optimal solutions for the optimization variables considered (bandwidth allocation of ground devices, task offloading time, local computing time allocation, and UAV trajectory) at each time slot. Extensive simulation experiments were conducted to demonstrate the performance improvement attained by the proposed BCD-based solution.

Overall, the proposed solutions discussed in this section suggest that UAV-based edge computing systems have certain advantages over cloud-based techniques in terms of optimization, convergence speed, throughput, and energy efficiency. These advantages make UAV-based edge computing systems a promising solution for various applications, including precision agriculture, smart cities, and disaster management, where real-time data processing and optimization are critical.

1.3.2. Summary of Related Works

A summary of the current literature is provided in Table 1. Onboard AI edge computing is becoming increasingly important for UAV systems, especially those utilizing EMC-based solutions. While EMC-based UAV systems offer benefits such as flexibility, resilience, and swift deployment, they also present new challenges that can only be addressed by advanced AI-based solutions, such as reinforcement and deep learning frameworks.

One reason for why onboard AI edge computing is necessary for EMC-based UAV systems is the need for real-time decision making. In certain applications, such as emergency response, decisions need to be made quickly and accurately. Onboard AI edge computing can process data in real time, allowing the UAV to make decisions based on the information that it collects, without the need for remote servers. This reduces latency and ensures that decisions are made in a timely manner.

Another reason is the need for autonomy. UAVs equipped with onboard AI edge computing can perform tasks autonomously, without human intervention. This is important in applications where it may be dangerous or impractical for humans to be present, such as in disaster response or surveillance missions. The AI algorithms on board the UAV can analyze the data collected and make decisions based on pre-defined rules, allowing the UAV to carry out its tasks independently.

Table 1. Comparative analysis of related work.

Reference	Scope	Advantages	Limitations
[18]	Overview of current applications of UAVs in remote sensing (up to 2015)	Comprehensive review	Limited to remote sensing domain
[19]	Discussion of UAV usage in 3D mapping (up to 2014)	In-depth coverage of UAV applications	Limited to 3D mapping applications
[20]	Covers UAV-based platforms for glaciology investigations (up to 2016)	Detailed discussion of UAV systems in glaciology	Covers only one domain application (glaciology)
[21]	Review of UAV deployments in forestry (up to 2017)	In-depth analysis of UAV systems in forestry	Restricted to European systems
[22]	Review of UAV applications in remote sensing (up to 2019)	Discusses multi-sensor fusion	Imbalanced coverage of UAV sub-systems
[23]	Edge computing usage in UAV visual communication	Extensive simulation of proposed framework	Lack of detailed comparison with existing frameworks
[24]	Path planning algorithm using RL paradigms	Optimal planning and routing decisions	Does not analyze the stability of the RL-based policies
[26]	Autonomous trajectory planning for UAV missions	Use of 5G wireless infrastructure	Lack of comparison with existing SOTA solutions
[27]	Multi-agent Q-learning algorithm for energy optimization	Higher energy consumption efficiency in UAV systems	Does not discuss the tradeoffs required to achieve energy efficiency
[28]	A new solution for MTU association and UAV trajectory.	Efficient RL-based solution using DDQN algorithm	Missing analysis of the DDQN limitations
[29]	Joint optimization of UAV cruise control and task offloading allocation	Efficient advantage actor–critic (A2C) solution	No comparison with SOTA policy gradient algorithms
[30]	Path planning using Monte Carlo tree search (MCTS) algorithm	Constraint-based maximization of average throughput	Benchmarking with relatively basic RL-based solutions
[31]	Battery and computation resource optimization using fog computing solutions	Improved UAV usage and reduced average delay	Missing comparison with SOTA solutions
[32]	UAV energy efficiency using multi-stage stochastic optimization	Optimal resource allocation during UAV movement	Higher computational complexity
[33]	Constraint-based joint optimization of bandwidth allocation, task offloading time allocation, and UAV trajectory	Joint optimization using block coordinate descent (BCD) algorithm	Lack of in-depth comparison with SOTA solutions

Furthermore, onboard AI edge computing allows for a more efficient use of resources. With the computing power on board, data can be processed locally without the need for constant data transmission to remote servers. This saves time and energy, and allows for a more efficient use of the UAV's limited resources, such as its battery life.

Based on the previous review of the existing literature, there is a growing trend in adopting EMC-based UAV systems, given their flexibility, resilience, and swift deployment. However, new challenges emerge with the deployment of such systems that can be handled only by advanced AI-based solutions, including reinforcement and deep learn-

ing frameworks. In fact, the solutions reviewed in the previous section are founded on well-established algorithms that have shown promising results in other engineering and science fields, including the optimal policy for emergency situations, data fusion, and information retrieval [34–36].

UAVs are becoming increasingly prevalent across multiple industries due to their flexibility and resilience. MEC-enabled UAVs are capable of providing computing and communication services at the network edges, even for ground-based units in areas with limited network coverage. This is particularly important in the field of remote sensing, where data collected from sensors on board the UAV need to be processed and analyzed in real time to support timely decision making. The ability of MEC-equipped UAVs to handle computing tasks and communication services at the network edges can significantly improve the speed and accuracy of remote sensing data collection processes.

Adopting edge computing for the onboard processing on UAVs is a challenging problem, yet beneficial from several perspectives. Embedding computation-intensive applications on the UAV edge device requires sufficient energy, storage, and computation resources to manage the demanding requirements of AI tasks. However, with the evolution of edge devices' capabilities, most of these challenges are overcome to a large extent, which makes edge computing in UAVs possible.

2. Materials and Methods

2.1. The AERO System

2.1.1. Why AI-Enabled Edge Computing for UAVs?

AI-enabled edge computing for UAVs can provide several benefits, including a low latency, increased efficiency, improved reliability, and enhanced privacy, as described below.

1. Low Latency: with advances in graphics processing units (GPUs) for edge devices (e.g., NVIDIA's Jetson boards), edge computing enabled the real-time processing of AI tasks, such as object detection, recognition, and tracking. This was not possible a couple of years ago. Consequently, edge computing promotes the real-time processing of data on board by allowing the drone to make quick local decisions about detected objects (e.g., the detection of a person to rescue) before sending the information to the cloud, thus saving useless communication with the server.
2. Increased efficiency: this approach also improves efficiency by decreasing communication overhead, saving bandwidth usage, and reducing the latency and load of the cloud servers. In fact, in the case of the cloud computing approach, the drone has to stream images at a high frequency and offload AI computation to the cloud. This is greedy in terms of the bandwidth and communication overhead, induces more communication latencies, and lacks scalability and computation cost, as the cloud cannot tolerate massive video traffic with real-time data processing. Edge computing helps to reduce the amount of data to be transmitted over a network and sent to the server.
3. Improved Reliability: computation on edge also improves the reliability of AI-based UAV applications. First, the drone data collection process will be less affected by the possible loss of communication due to the increased autonomy of the drone by locally processing collected data. In case of total communication loss, the data of detected objects are still saved locally and transferred to the cloud when the communication is back or offline in the worst scenario. In addition, edge computing makes the processing of AI tasks distributed among the UAVs and not centralized in the cloud, which can be vulnerable to outages or other disruptions. There are two resulting benefits: (1) it avoids the single point of failure, and (2) it increases the system's scalability as computing is fully distributed.
4. Better privacy: the local processing of collected images and detected objects helps to enhance privacy preserving by reducing the amount of data that are transmitted and stored in centralized remote servers. Adopting strong encryption on individual detected object frames is more efficient than encrypting the whole video stream. In

addition, collected object images transmitted to the cloud will remain private and secure against unauthorized access, as they no longer require being processed as plain data.

2.1.2. AERO System Architecture

In this section, we present the system architecture of AERO, shown in Figure 1.

The objective of the AERO system is to provide an ecosystem for using an edge-device on UAVs to execute complex deep learning algorithms to help automate computer vision applications, including object detection and tracking, on board the UAVs.

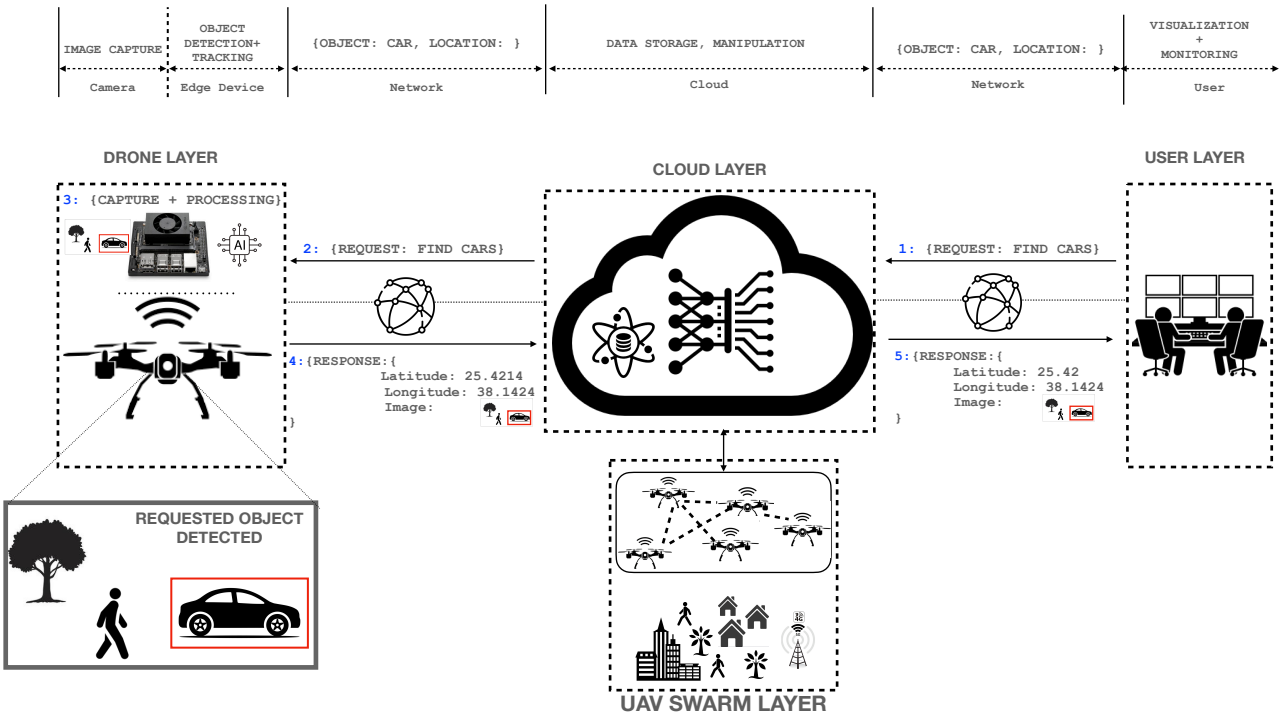

Figure 1. AERO system architecture.

The AERO system is composed of four layers:

- The Drone Layer: this represents the one UAV subsystem that is equipped with onboard processing and storage capabilities to perform AI tasks such as image and video analysis in real time. Edge computing is used to locally process collected raw data rather than sending them to a remote server as a video stream. In the UAV AERO, the edge device is a GPU-based embedded system (e.g., NVIDIA Jetson Xavier board) directly attached to the drone's camera through a proper channel (USB port, Ethernet (RTSP), or serial). The drone uses its network interfaces (e.g., 4G/5G cellular networks or WiFi) to communicate with and transmit detected objects' images to the cloud.
- The Swarm Layer: this layer consists of a cluster of UAVs equipped with camera sensors and AI-edge devices that coordinate together to perform a cooperative mission; for instance, distribute a search for lost people in a large area. In Figure 1, the UAVs swarm communicates with the cloud, which orchestrates their mission, rather than adopting ad hoc communication among the drones. The reasons are as follows:
  - Increased Reliability: the communication of UAVs with the cloud through cellular networks provides a more robust and stable connectivity compared to ad hoc swarms, which may be subject to interference and non-guaranteed message exchange, particularly in large-scale deployment. In critical applications such as search and rescue, it is essential to maintain reliable communication to ensure better coordination between drones through the cloud server.

  - Interference: in ad hoc swarm communication, the drones have to contend for channel access (e.g., CSMA/CA). This will lead to interference and collision, which requires message retransmissions. This results in poor communication efficiency and increased delays. Other approaches involve the use of time synchronization (e.g., time division multiple access (TDMA))), but these techniques are challenging as they need to maintain synchronization among the UAVs. Clock drift, latency, interference, and the dynamic nature of the UAVs can all impact the accuracy of the transmissions, leading to disruptions in the synchrony of the TDMA system.
  - Global Knowledge: with all swarm UAVs communicating with the cloud, the latter maintains up-to-date information about all UAVs, including their positions, their states, and the list of detected objects. The cloud can make informed decisions in real time and an adjustment of the mission plan or resource allocations. For example, if a UAV experiences low battery levels, the cloud will be better positioned to reassign its tasks to other drones based on optimized criteria. The cloud can also optimize the task allocation among all drones and give its global knowledge to ensure that mission execution is completed effectively.

  Overall, these planes work together to support the operation and management of a fleet of drones. The data plane handles the collection and processing of data, the user plane enables human users to interact with the system, and the drone plane manages the operation of the drones themselves.
- The Cloud Layer: as the UAV edge device performs AI computation-intensive tasks, the cloud system does not require having sextensive/advanced computing resources (GPU-based cloud systems are not required), which reduces the deployment cost considerably, as GPU-based cloud systems tend to be more expensive than CPU-based cloud systems. The cloud is responsible for data storage, manipulation, and visualization. The cloud is organized into three planes.
  - UAV Plane: the UAV plane is primarily responsible for managing the operation of a fleet of drones, including overseeing and coordinating the drones' activities, managing the data collected by the drones, and performing mission planning to ensure compliance and safety. The fleet management system (FMS) plays a critical role in controlling and monitoring drones, scheduling their tasks and missions, and ensuring their compliance with airspace regulations. These benefits include improved efficiency, data management, and safety.
  - Data Plane: the data plane is responsible for handling the large amounts of data generated by the drones' sensors and onboard equipment. During operation, the drones collect a large amount of data and send them to the cloud for storage and processing using advanced data analytics frameworks, and visualize dashboards to end-users for quick analysis and decision making based on the data collected by the drones. The data plane also ensures the persistence and availability of the data when needed by the end users through replication, caching, and load balancing.
  - User Plane: the user plane in the AERO system is responsible for interacting with users, including mission planning, monitoring, and control. It allows users to access the system through various interfaces and applications, such as a web-based dashboard, mobile app, or API. Through the user plane, users can create and manage drone missions, view real-time drone data, and receive alerts and notifications. Users can monitor the status and performance of the operating drones in real time, providing important information such as flight paths, battery levels, and sensor data. This feature is essential in situations such as emergency response scenarios and surveillance operations. The user plane is a critical component of the AERO system, enabling efficient and effective drone operations by providing a user-friendly interface for mission management and real-time monitoring.
- The End-User Layer: the end-user layer in the AERO system enables end-users to access the system through the Internet using web service APIs. The end-users use

interactive dashboards to monitor the status of their drones in real time, send commands, and receive real-time video streams that have been processed by deep learning applications located either at the edge or on the cloud. The end-user layer interacts with the cloud layer through its user plane, which provides access to authorized cloud resources and allows them to interact, monitor, and control drones for operation. The end-users can be of different types depending on their role.

- Authority: responsible for authorizing drone operations, managing the drone fleet, and ensuring compliance with regulations.
- Operator: responsible for managing and operating drone fleets, executing drone missions, and ensuring safety.
- User: requests drone operations for various purposes, such as aerial photography, surveying, or inspection.

2.1.3. AI-Enabled UAV

This section describes the UAV platform that we used to test the AERO system in practice. Figure 2 depicts our custom-built battery-powered hexacopter platform and highlights its main components. The hexacopter specifications are detailed in Table 2.

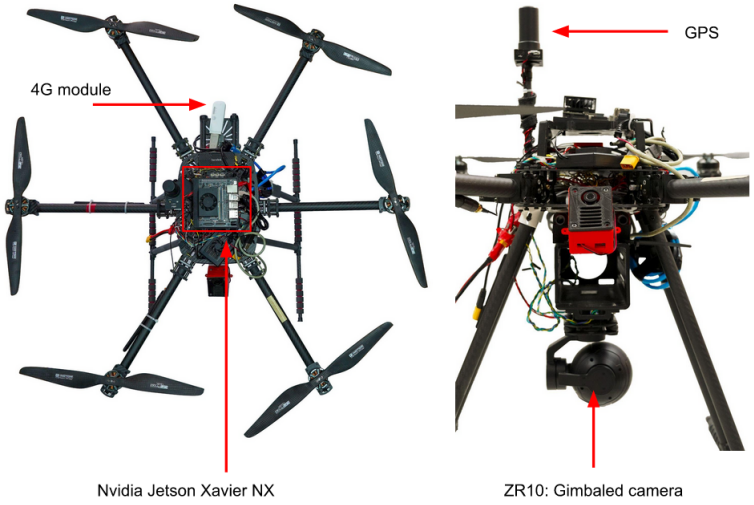

Figure 2. Top view of the custom hexacopter.

Table 2. Hexacopter specifications.

Number of motors	6
Motor type	T-Motor MN3508 KV380
Propeller size	15″
Wheelbase	850 mm
Battery	6200 mAh
Maximum takeoff weight	7 kg
Maximum flight time	15 min
Camera	ZR10 (30× zoom, 2K resolution)
Edge device	NVIDIA Xavier NX
Limited communication range	15 Km (2.4 Ghz)
Extended communication range	using 4/5G networks

The selected hexacopter platform was equipped with custom onboard electronics to enable edge computing as well as continuous cloud connectivity. The hardware architecture of the custom onboard electronics and communication systems are shown in Figure 3, and are described as follows.

- Gimbal–camera System: this is a camera–gimbal system which consists of the main vision sensor that is stabilized by a 3-axis gimbal. This system is called a SIYI ZR10

gimbal–camera system and has a 30× hybrid zoom (10× optical and 3× digital) and a 2K camera. The gimbal–camera system has its own microprocessor, which has an RTSP (real time streaming protocol) server that sends real-time image streams to clients (edge and communication devices) using Ethernet connections. In addition, the camera orientation is stabilized and controlled by a 3-axis gimbal to control the visual region of interest during flight.

- NVIDIA Jetson Xavier NX: this is the main computation board (edge device) and has adequate GPU power to perform real-time object detection and advanced autonomous surveillance mission planning. It is connected to the camera–gimbal system, via an Ethernet switch, to receive the real-time image stream and send camera–gimbal commands to control the camera orientation and zoom level. The Xavier NX runs our custom software, which performs real-time object detection and localization, which is described in Section 2.2. It also has a connected 4G module to enable extended communication with the cloud server to send information about the detected objects and receive surveillance mission requests.
- 4/5G communication: a 4/5G communication module is connected to the Xavier NX module to enable communication with the cloud server for an extended range. The communicated information includes the image frames with metadata (e.g., detected objects and their coordinates) sent from the edge device to the cloud server, and mission requests from the cloud server to the edge device.
- Ethernet switch: this hardware module is used to allowfor transmitting the camera image stream to the onboard computer (Jetson Xavier NX) for image processing, as well as the air unit transceiver, which communicates with a ground remote controller for visualization.
- Pixhawk Orange Cube flight controller: this is the autopilot hardware, which runs the well-known open-source PX4 autopilot firmware [37]. The autopilot stabilizes the drone's position and executes planned missions that are sent by the onboard computer.
- Air unit transceiver: this module exchanges image streams and UAV telemetry with a ground remote controller using a 2.4 GHz link.
- Remote controller: the ground remote controller is used by the UAV backup pilot to control the drone maneuvers, if needed, and have real-time visual feedback of the onboard camera stream.

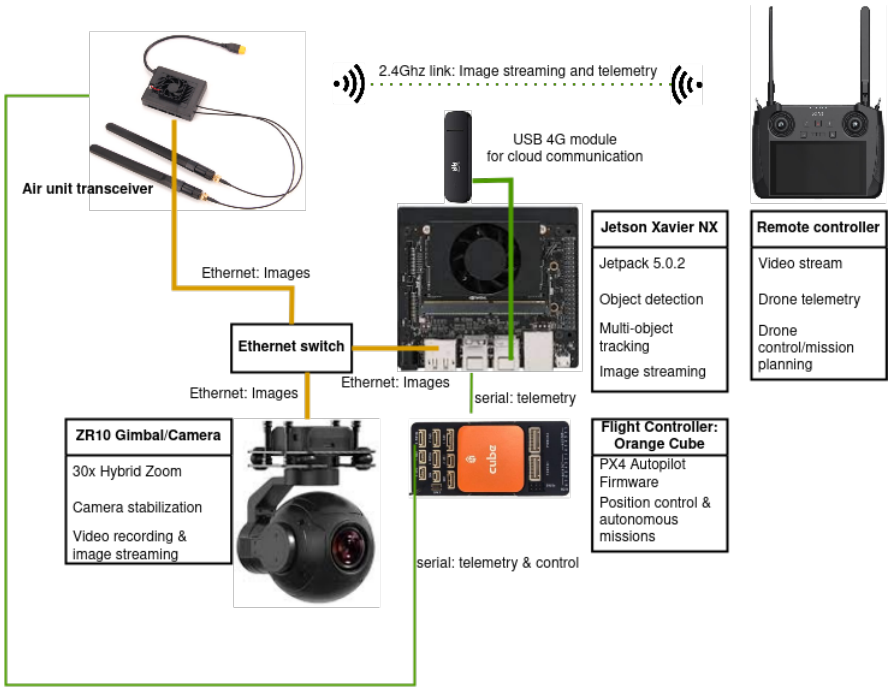

Figure 3. UAV onboard electronics.

2.2. The AERO AI Module

In this section, we present the AERO brain system that leverages YOLOv7 [38] for object detection, DeepSort [39] for object tracking, and TensorRT (TRT) [40] acceleration to ensure the real-time execution of the model on edge devices. The novelty of our approach is the design of a multi-stage deep learning model that allows for making object inferences over several consecutive frames to optimize the detection performance in two main aspects:

- Accuracy: typical object detection and tracking models perform inference on one static image from the video frame, which usually leads to high misclassification ratios. We dramatically improved the accuracy by considering several consecutive frames and using a voting approach to maximize the object recognition accuracy.
- Real Time: a multi-stage model uses several deep learning models in sequence. The deployment of a multi-stage model makes real-time inference more challenging, particularly on embedded edge devices, considering their lower capabilities. We overcame this issue by using TensorRT acceleration on NVIDIA's Jetson AXG to maintain a high frame rate for the AERO multi-stage inference model.

2.2.1. AERO Model Architecture

Figure 4 shows the main steps of the processing performed by the AERO AI module on edge. The AERO model is composed of three modules, namely the Detection Module, the Model Acceleration Module, and the Tracking Module, described as follows.

Detection Module

The detection module is based on YOLOv7, which is the latest version of the widely used YOLO family of single-stage object detectors. It established the state of the art both in terms of accuracy and speed, outperforming competitor models by a large margin. For comparison, we also tested YOLOv4 [41], which is still one of the most popular object detection models.

The DeepSORT tracker is an extension of the simple online and real-time tracking (SORT) algorithm [42], which is an efficient algorithm used for real-time object tracking. The key innovation of DeepSORT is the incorporation of a pre-trained deep association metric that utilizes object appearance information to improve the tracking performance. The deep association metric in DeepSORT uses a pre-trained deep neural network to encode the appearance information of objects. By comparing the features extracted from the neural network, DeepSORT is able to estimate the likelihood of two objects being the same. This allows DeepSORT to handle challenging scenarios such as occlusion, appearance changes, and the temporary disappearance of objects. Overall, DeepSORT provides a robust and accurate solution for tracking multiple objects in real time. Its ability to incorporate appearance information allows it to handle various challenging scenarios, making it an ideal solution for applications such as surveillance, robotics, and autonomous vehicles. For these reasons, and for its popularity in the literature, we opted for this particular tracker, although any other multi-object tracker could be used in our system. To integrate DeepSORT with the YOLO object detector and the other components of our system, we modified the implementation of the track class in a similar way to the one described in [43].

The object detection and tracking system processes each new frame by first applying YOLOv7 on the entire frame to obtain bounding boxes and confidence scores for all detected objects. These bounding boxes are then input to DeepSort, the multi-object tracker, which produces pairs of matched tracks and detections as well as lists of unmatched tracks and detections. For each track, the system checks whether it should be discarded, further processed, or sent to the server.

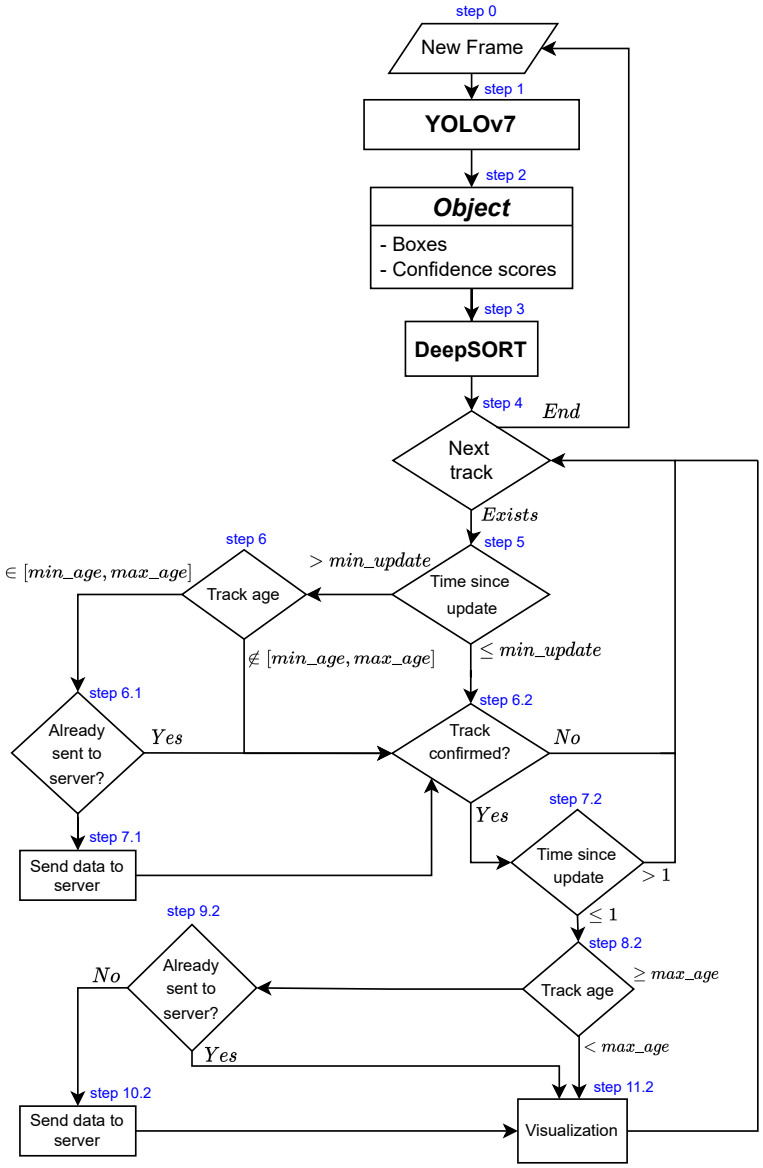

Figure 4. The AERO model architecture of the AI module.

First, the system checks if the track has not been matched with a detected bounding box for more than a predefined number of consecutive frames (default value of 10). If so, the system assumes that the object is no longer in the camera's field of view. Next, the system checks if the track's age (number of frames in which the same object has been detected) is within a predefined interval (default value of [2, 40]). A low value indicates that the track is unreliable, whereas a high value means that the object information has already been sent to the server. The default values of the minimum number of consecutive frames and the track's age interval were fixed empirically after a series of preliminary tests.

In all cases, the system checks if the current track has been confirmed by being observed in the required minimum number of consecutive frames (default value of 3) and has not been deleted due to missed detections. If the track is confirmed or has been matched with detected bounding boxes in the current or previous frames, the system checks its tracking age. If the age is equal to or greater than the maximum allowed age, the system sends its information to the server if it has not yet been sent. Finally, the system can optionally visualize the object's bounding box and information using the current attributes of the track instance.

If the track is not confirmed or has not been matched with bounding boxes for at least two consecutive frames, the system skips it and moves on to the next track. By following this process, the object detection and tracking system can accurately detect and track objects in real time while minimizing false detections and conserving computational resources.

Model Acceleration Module

While deep learning models can provide highly accurate results, they require significant computational and storage resources to train and run, even for YOLOv7, which is the fastest object detector to date. This makes deploying deep learning models on edge devices such as Jetson boards a challenging task as these devices often have limited resources in terms of memory and processing power.

To address this challenge, we leveraged the use of the TensorRT acceleration framework. TensorRT is a high-performance inference engine developed by NVIDIA that allows developers to optimize deep learning models for deployment on a range of NVIDIA platforms, including Jetson edge devices. It can optimize models by reducing the precision of model parameters and minimizing the memory required to store them, allowing the model to run more efficiently on edge devices with limited resources. TensorRT can also optimize models by using dynamic tensor memory allocation, which allocates memory dynamically during inference, reducing the overall memory usage.

The TensorRT optimization framework also optimizes models by fusing layers, which combines multiple layers in a neural network into a single layer to speed up model inference. This is particularly important for applications that require real-time processing on edge devices, where latency is critical, such as real-time surveillance applications. In a previous study [44], we have shown that TensorRT optimization provides the fastest execution on a wide variety of cloud and edge devices. This demonstrates the effectiveness of TensorRT in optimizing deep learning models for edge devices, achieving faster inference times and a lower latency.

Target Localization Module

In [2], we proposed a methodology for object detection and location estimation based on established photogrammetry concepts and metadata extracted from drone images, including EXIF and XMP data. This approach allows for accurately estimating the GPS location of detected objects within each frame. The use of metadata, such as the drone's altitude and GPS location, image size, and calibrated focal length, provides a demonstrably sound basis for determining the location of objects in the images.

To account for potential errors or uncertainty in the distance estimation, the algorithm also incorporates a correction factor based on the ratio between the drone's altitude and the estimated average height of the objects using the formula:

$$\begin{cases} D_x^c = \left(1 - \frac{h}{H}\right) D_x \\ D_y^c = \left(1 - \frac{h}{H}\right) D_y \end{cases} \quad (1)$$

where:

- D_x and D_y are the coordinates of the object's bounding box center before correction.
- D_x^c and D_y^c are the object's coordinates after correction.
- h is the estimated average object height.
- H is the drone altitude.

Additionally, the algorithm considers the yaw degree of the image to refine the location estimation of each object further. This approach allows for an accurate counting of objects even when there are overlaps between images, further demonstrating the scientific rigor of the methodology. This same methodology can be applied to the detected objects in the AERO system, although we did not include this target localization module in the experimental part of the current study.

3. Results

3.1. Experimental Setup

For the experimental evaluation, we tested two different object detection models (YOLOv4 and YOLOv7), two different implementations (PyTorch and TensorRT), three different video resolutions (1920 × 1080 for 2 videos, 2688 × 1512, and 3840 × 2160, see Figures 5 and 6), and three different devices (RTX8000, Jetson Xavier AGX, and Jetson Xavier NX). The videos' length ranges from 0.5 mn to 5.9 mn. Videos 1 and 3 were used for the detection of six classes of objects (car, person, bicycle, bus, monocycle, and truck), whereas videos 2 and 4 were used for the detection of a single class (car). On top of each bounding box, information is displayed about the detection class, the tracking ID, the number of frames in which the same object has been observed, and the object color. For videos 1 and 3, the number of objects of each class is also displayed on the top left corner. The outputs of videos 2 and 4 are available on this link: shorturl.at/nrzOY (accessed on 30 March 2023). As for videos 1 and 3, the original footage was provided by a third party that did not agree to disclose them.

Table 3 presents the conducted experiments that are analyzed below. Due to software environment limitations and compatibility issues, some frameworks did not work on some devices. We were able to run all configurations on Jetson Xavier NX (Jetson pack 5, TensortRT 8), whereas YOLOv7 did not work on Jetson Xavier AGX (Jetson pack 4.5, TensorRT 7) and the TRT versions of YOLOv4 and YOLOv7 did not work on the RTX8000 GPU (CUDA version 10.0).

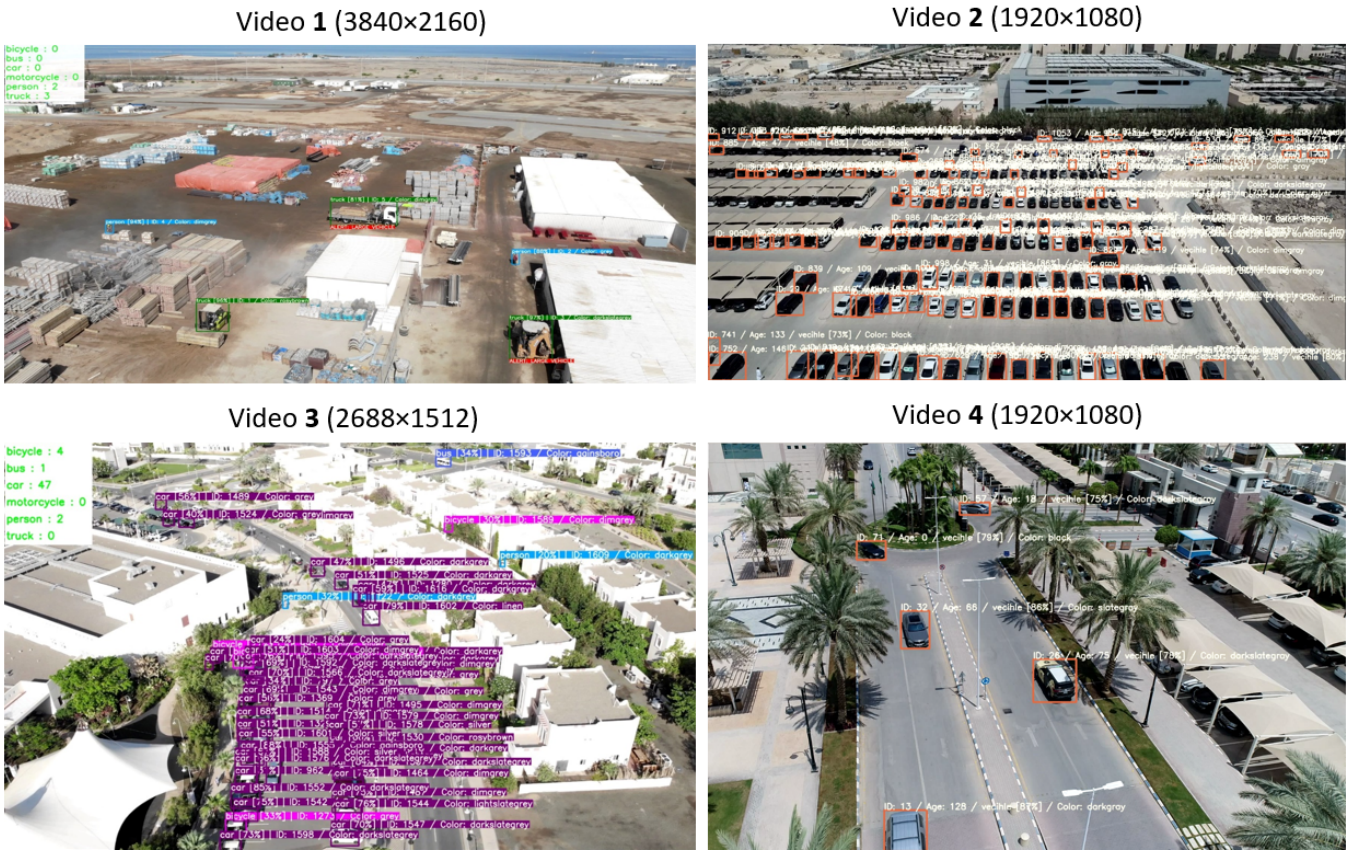

Figure 5. Sample frames from the output of the four videos used for the evaluation of the AI-enabled system, with different resolutions.

We chose the YOLOv7 object detector because it was the state-of-the-art object detector in terms of accuracy and speed at the time of this study. As for YOLOv4, we tested it for comparison, seeing that it is still one of the most popular object detectors (YOLOv5 and YOLOv6 are not as popular in the literature). For our case study, we could not use

the pre-trained models of YOLOv4 and YOLOv7 because they were mainly trained on ground-level images (COCO dataset or OpenImages dataset), and we are dealing with aerial images. Consequently, for training YOLOv7, we used the VisDrone dataset [45], which we filtered to keep only one class of vehicles (cars), and, for YOLOv4, we trained a model on a private dataset containing 940 UAV images showing six classes (car, person, bicycle, bus, monocycle, and truck) with a total of 33,088 instances. These images were captured in the Jeddah region in Saudi Arabia, in daylight and sunny conditions, and were manually labeled. Table 4 summarizes the main hyperparameters and results of the training of the YOLOv4 and YOLOv7 object detectors. Since we built our custom dataset gradually, we show the results of the training for several sizes of the dataset. We observe that there is a stagnation in terms of the mAP (mean average precision) when moving from 545 to 821 training images. YOLOv7 shows notably better results in terms of mAP but they are not directly comparable to YOLOv4's results since the number of classes is different.

Figure 6. Close view of a sample of the output videos showing various classes, and some false negative and false positive detections.

Table 3. Conducted experiments on different devices, object detection models, frameworks, and input video resolutions.

			Device		
Model	**Implementation**	**Resolution**	**RTX8000**	**Jetson Xavier AGX**	**Jetson Xavier NX**
YOLOv4	TRT	2688 × 1512		✓	✓
		3840 × 2160		✓	✓
YOLOv7	TRT	1920 × 1080			✓
	PyTorch	1920 × 1080	✓		✓

Table 4. Hyperparameters and results of the training of the YOLOv4 object detector for several configurations.

YOLO Version	Dataset	Nb Classes	Training Images	Validation Images	Input Size	Best mAP
v4	Custom	6	311	35	608 × 608	41.9%
v4	Custom	6	545	60	768 × 768	57.0%
v4	Custom	6	821	91	768 × 768	57.0%
v7	VisDrone	1	4935	617	640 × 640	91.3%

3.2. Performance Evaluation

We first analyzed the inference speed for each device and detection model using a series of box plots (Figures 7–9). Box plots are a useful way to visualize the distribution of data and compare data across multiple variables, and can provide insights into the central tendency, variability, and skewness of the data. The grey line inside each box represents the median value of the data. Half of the data points fall above this line and half fall below. The box itself represents the interquartile range (IQR), which contains the middle 50% of the data. The bottom of the box represents the first quartile (Q1), or the value at which 25% of the data fall below. The top of the box represents the third quartile (Q3), or the value at which 75% of the data fall below. The whiskers extend from the box to show the range of the data, excluding any outliers, while the individual blue points represent a 1D scatter plot of the data.

Figure 7 depicts the box plot of the inference speed in frames per second (FPS) for each device, detection model, and input video resolution. We observe that the TensorRT optimization of the YOLOv4 model provides the fastest inference speed, even on higher-resolution input videos, whereas, for YOLOv7, the TRT optimization provides no gain in speed. In contrast, the average inference speed deteriorates from 7.2 FPS (for the PyTorch implementation) to 2.8 FPS (for TRT). This is likely due to the fact that the new features introduced in YOLOv7 are not yet adequately optimized in the latest versions of TensorRT.

Figure 8 shows the box plot of the inference speed for each device and detection model in the case where the detected objects are sent to the cloud and in the case where the connection to the cloud is disabled. In all cases, the connection to the cloud significantly slows down the inference speed of the whole system. The average speeds drops from 12.3 FPS when no data are sent to the cloud to 5.0 FPS when sending data to the cloud. This highlights the importance of choosing a high-quality network and optimizing the edge–cloud communication.

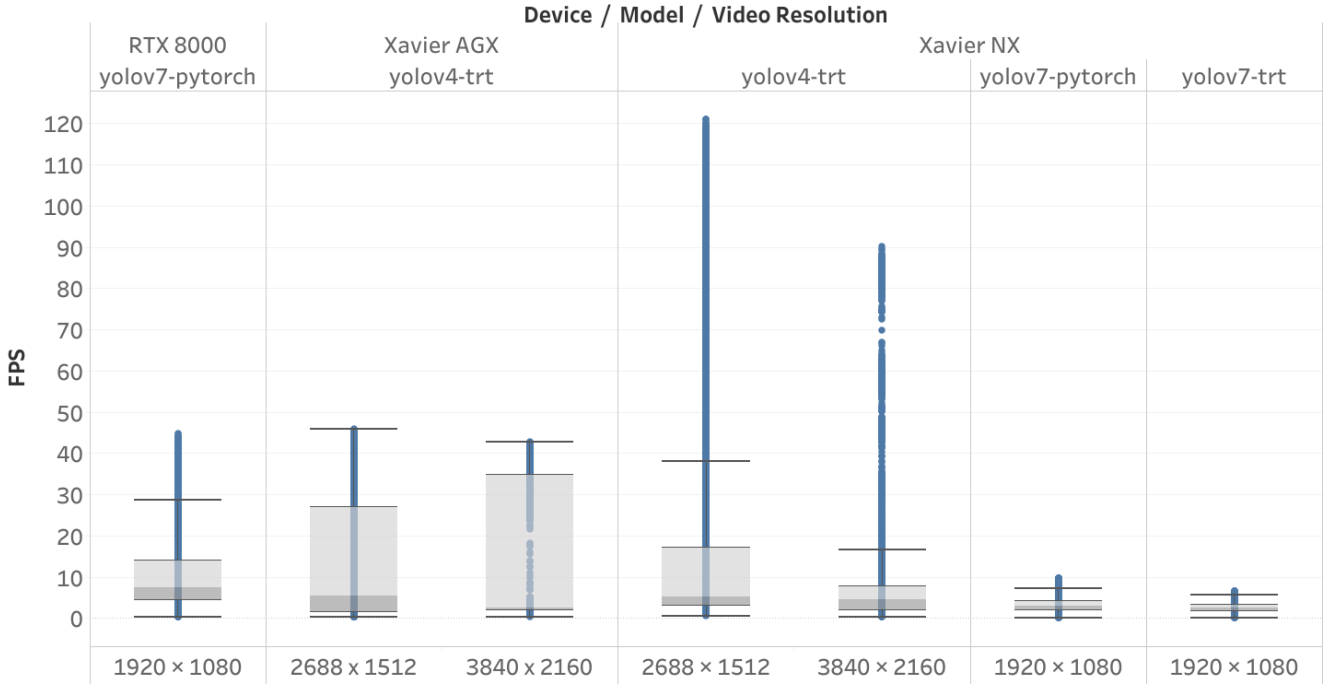

Figure 7. Inference speed per device, detection model, and video resolution.

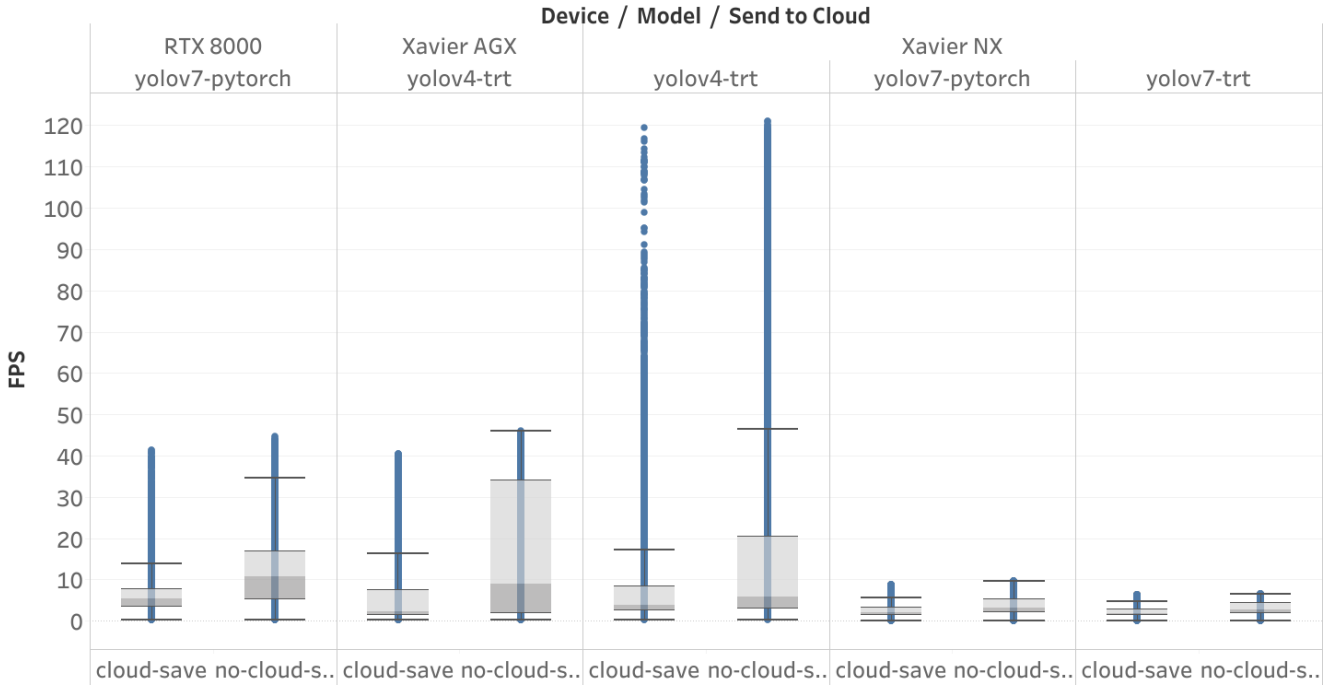

Figure 8. Inference speed per device, detection model, and connection to the cloud.

Figure 9 shows the box plot of the inference speed for each device and detection model in the case where the DeepSORT tracker is included or excluded. On all devices, and for all object detection models, the use of the tracker markedly decelerates the system. The average inference speed declines from 19.6 FPS (without tracker) to 5.0 FPS (with tracker). Nevertheless, the use of the tracker is necessary to correctly count the number of objects and send each object's information to the server only once. We should, however, investigate faster multi-object trackers to enhance the overall system speed.

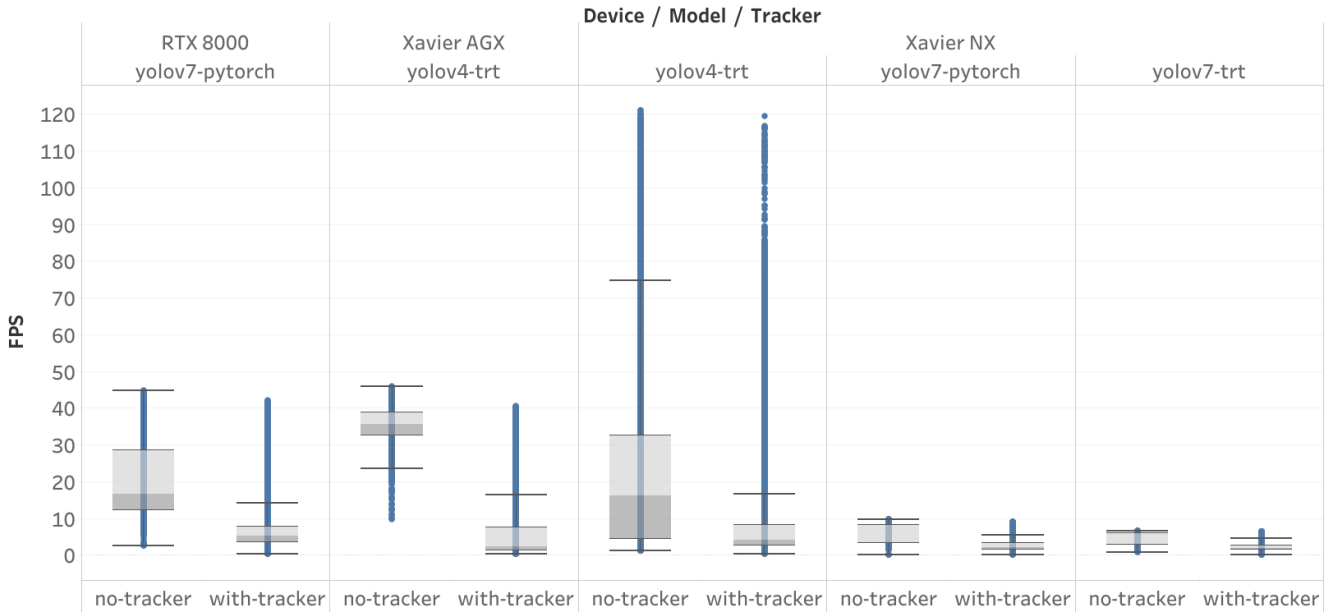

Figure 9. Inference speed per device, detection model, and use of tracker.

To analyze the influence of each component of the AI system and control for variability due to different devices and video resolutions, we generated a set of scatter plots to measure the inference speed on the Jetson Xavier NX device with an input video resolution of 1920 × 1080. Figure 10 illustrates the scatter plot of the inference speed per number of detected objects in each frame using both PyTorch and TRT versions of the YOLOv7 object detection model. As previously noted (about Figure 7), the PyTorch implementation achieved higher inference speeds compared to the TRT implementation. Figure 10 appears as a superimposition of three plots, which we will distinguish in subsequent figures.

Figure 10. Inference speed (in FPS) per number of detected objects, on Jetson Xavier NX, with an input video resolution of 1920 × 1080, using the PyTorch and TRT version of the YOLOv7 object detection model.

Figure 11 presents the scatter plot of the inference speed per number of detected objects, on Jetson Xavier NX, with an input video resolution of 1920 × 1080, using the TRT version of the YOLOv7 object detection model, when including or excluding the tracker. As already noted in Figure 9, the use of the tracker significantly slows down the system performance. The blue dots in Figure 11 represent the measures that included the tracker, and correspond to the lower part of the plot in Figure 10. The magenta dots, corresponding to the inclusion of the tracker in the AI system, still appear as the superimposition of two plots. They will be distinguished in the next figure.

Figure 12 shows a similar scatter plot but with no tracker when including or excluding the local saving of the output video. It demonstrates that storing the resulting output video on the edge's disk consumes a significant amount of time and markedly slows down the overall inference speed. The system speed decreases from 12.9 FPS to 6.8 FPS on average over all devices and configurations. For the configuration shown in Figure 12 (Jetson Xavier NX, YOLOv7 TRT, no tracker, 1920 × 1080 video resolution), the average inference speed drops from 5.8 FPS to 2.7 FPS when saving the output video. Consequently, this local storage should not be used unless it is absolutely required for the application.

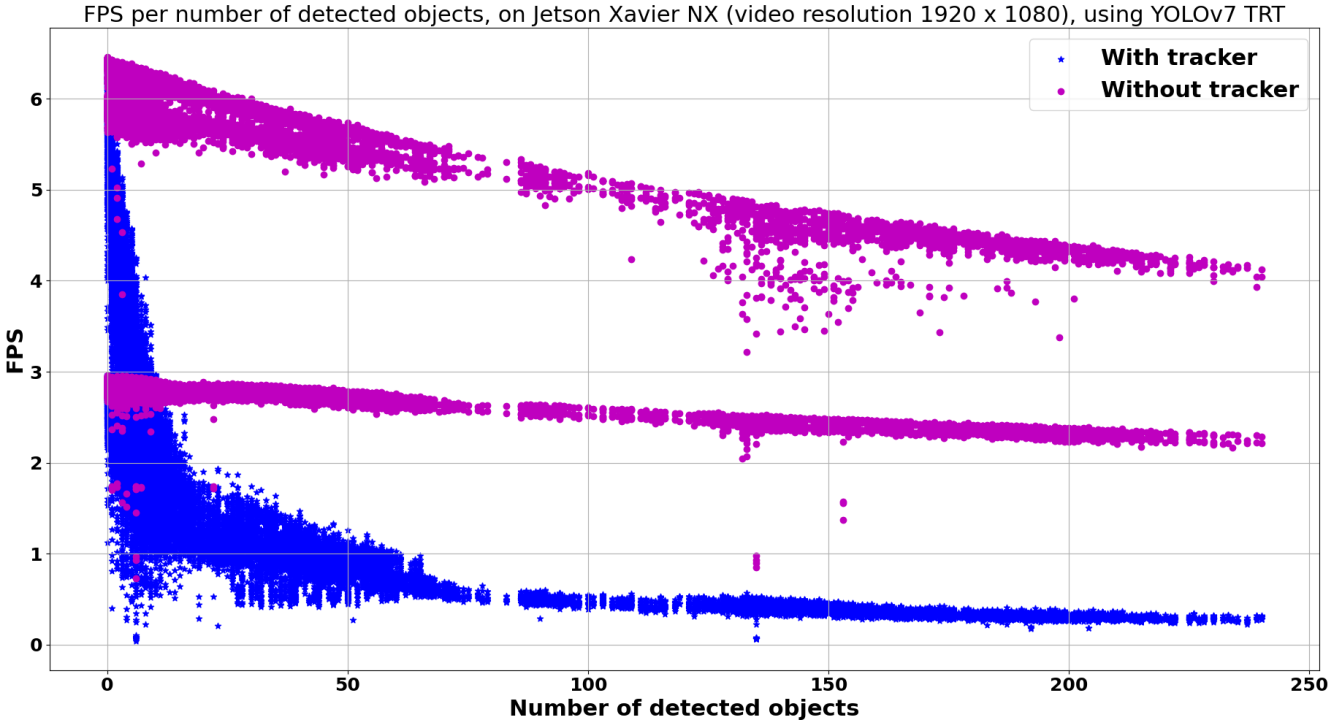

Figure 11. Inference speed (in FPS) per number of detected objects, on Jetson Xavier NX, with an input video resolution of 1920 × 1080, using the TRT version of the YOLOv7 object detection model, when including or excluding the tracker.

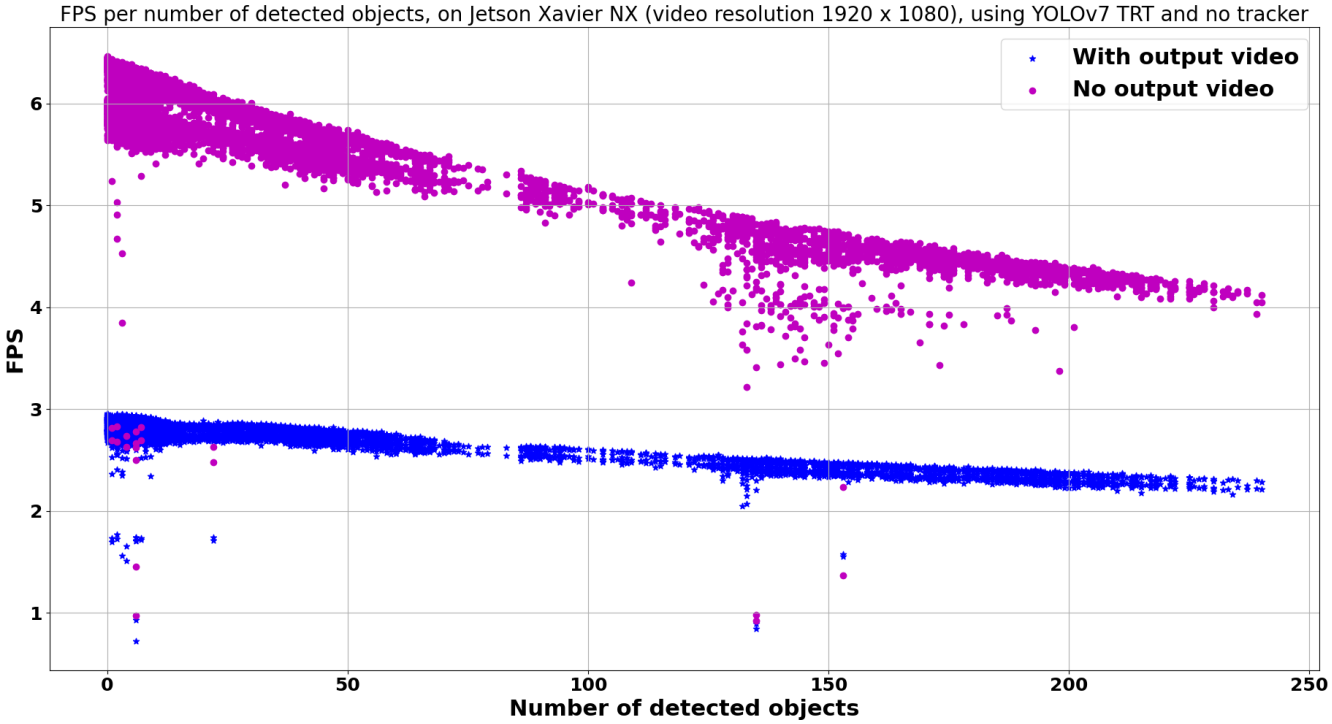

Figure 12. Inference speed (in FPS) per number of detected objects, on Jetson Xavier NX, with an input video resolution of 1920 × 1080, using the TRT version of the YOLOv7 object detection model, with no tracker, when including or excluding the saving of the video output.

4. Discussion

From Figures 7–12, we conclude that the inference speed of an AI system for object detection can be affected by various factors, including the device used, the detection model, the input video resolution, the use of cloud connectivity, and the inclusion of a tracker or local saving of output videos. The TensorRT optimization of the YOLOv4 model provides the fastest inference speed even on higher-resolution input videos. However, for YOLOv7, the TRT optimization did not provide any gain in speed due to an inadequate optimization of new features in the TensorRT version used. Sending data to the cloud significantly slows down the inference speed, highlighting the importance of choosing a high-quality network and optimizing edge–cloud communication. The use of a multi-object tracker is necessary to correctly count the number of objects and send each object's information to the server only once, but it markedly decelerates the system. Finally, avoiding the local saving of the output video can also help to improve the system's inference speed. Therefore, the best configuration for an AI system for object detection depends on the specific application requirements and hardware constraints.

To assess the accuracy of the object detector, the influence of the TRT optimization, and the multi-object tracker, we selected two test videos (see Figure 5):

- video 1: showing six classes (car, person, bicycle, bus, monocycle, and truck), with an average of six objects per frame, an input resolution of 3840 × 2160, a length of 50 s, and an FPS of 30.
- video 4: showing a single class of cars (with an average of six cars per frame), with an input resolution of 1920 × 1080, a length of 4 mn and 25 s, and an FPS of 24.

We manually counted the following metrics on still images extracted from the video every 20 frames (75 frames for video 1 and 319 frames for video 4):

- FP: number of false positives (objects incorrectly detected) generated by the object detection model.
- FN: number of false negatives (non-detected objects) generated by the object detection model.

- Precision: $Precision = \frac{TP}{TP+FP}$, where TP is the number of true positives (correctly detected objects).
- Recall: $Recall = \frac{TP}{TP+FN}$
- F1 score: $F1score = \frac{2 \times Precision \times Recall}{Precision+Recall}$
- Identity switches: number of switches between the IDs assigned by the tracker. This happens when the tracker conflates two objects that are too close.
- Identity changes: number of changes in the IDs assigned by the tracker to the same object. This happens when the tracker misinterprets a single moving object for two objects.

Table 5 summarizes the obtained results for these metrics when using the TRT implementations of the YOLOv4 object detector on video 1. The number of FNs is relatively low compared to the number of FPs due to the fact that most vehicles have a relatively large size (compared to video 4). The number of identity switches and changes is also reduced compared to video 4 because the distance between objects is markedly larger, which makes the tracker's task easier. Figure 13 shows two close frames from the output of video 1 where several detection and tracking errors appear. We notice one false positive in frame 240 ('person'), and two other false positives in frame 260 ('person' and 'truck'), as well as a misclassification (truck classified as 'person'). Between the two frames, there are three identity changes (4→34, 5→4, and 30→19). Identity switches often happen when two objects move close to each other, while identity changes may happen when the object's speed is relatively high.

Figure 13. Sample frames from the output of video 1, showing frames number 240 and 260.

On the other hand, Table 6 summarizes the obtained results when using the PyTorch or the TRT implementations of the YOLOv7 object detector on video 4. The difference between the two implementations is relatively minor, except for identity switches, which double from 5 to 10 when converting the PyTorch model to TRT. This indicates a loss in precision in the converted detection model that impacts the tracker accuracy. Nevertheless, this figure remains relatively low (1.6% to 3.1% relative to the number of frames) considering the number of cars and the duration of the video. By contrast, the number of identity changes is much higher, both for the PyTorch and the TRT implementations. The tradeoff between the number of identity switches and identity changes can be modified by changing the tracker hyperparameters, but we consider the identity switches to be more critical because they entail the conflation of the information of different objects, whereas the identity changes only result in duplicate information sent to the server. On the other hand, we observe that the number of false negatives is much higher than the number of false positives. In fact, small or occluded objects are often missed by the object detector, as can be seen in Figure 5. Consequently, the precision is high (99.3% for both PyTorch and TRT implementations), whereas the recall is much lower (72.5% and 73.1% for PyTorch and TRT, respectively). This tradeoff can also be modified by changing the score threshold for the object detector.

Table 5. Number of false positive detections (FPs), false negative detections (FNs), precision, recall, F1 score, identity switches, and identity changes for the TRT implementation of the YOLOv4 object detection model on test video 1 (resolution of 3840 × 2160, length of 50 s, FPS of 30) captured by a drone, showing 6 classes (car, person, bicycle, bus, monocycle, and truck).

	FP	FN	Precision	Recall	F1 Score	Identity Switches	Identity Changes
YOLOv4 TRT	80	33	82.7%	92.1%	87.1%	16	26

Table 6. Number of false positive detections (FPs), false negative detections (FNs), precision, recall, F1 score, identity switches, and identity changes for the PyTorch and TRT implementation of the YOLOv7 object detection model on test video 4 (resolution of 1920 × 1080, length of 4 mn and 25 s, FPS of 24) captured by a drone, showing a single class of 'cars'.

	FP	FN	Precision	Recall	F1 Score	Identity Switches	Identity Changes
YOLOv7 PyTorch	20	1136	99.3%	72.5%	83.8%	5	184
YOLOv7 TRT	22	1099	99.3%	73.1%	84.2%	10	176

5. Conclusions

The commercial usage of UAVs is still largely limited by the lack of onboard AI on the edge, leading to manual data observation and offline processing after data collection. Alternatively, some approaches rely on the cloud computation offloading of AI applications, which can be unscalable and infeasible due to a limited connectivity and high latency of remote cloud servers. To address these issues, in this paper, we proposed a new approach that uses edge computing in drones to enable extensive AI task processing on board UAVs for remote sensing applications. The proposed system architecture involves a cloud–edge hybrid approach where the edge is responsible for processing AI tasks and the cloud is responsible for data storage, manipulation, and visualization.

To implement this architecture, coined AERO, we designed a UAV brain system with onboard AI capabilities that uses GPU-enabled edge devices. AERO is a novel multi-stage deep learning module that combines object detection (YOLOv4 and YOLOv7) and tracking (DeepSort) with TensorRT accelerators to capture objects of interest with a high accuracy and transmit data to the cloud in real time without redundancy. AERO processes the detected objects over multiple consecutive frames to maximize detection accuracy. The

experiments show that the proposed approach is effective for utilizing UAVs equipped with onboard AI capabilities for remote sensing applications.

While the proposed system architecture and AERO module were designed to process visual data from UAVs, future work could explore the integration of other sensors, such as LiDAR or thermal cameras, to enhance the accuracy and efficiency of remote sensing applications. In addition, we plan to explore the integration of autonomous navigation capabilities to enable UAVs to navigate and collect data independently, without the need for manual control or intervention.

Another crucial aspect that needs to be considered in future works when designing drone systems with onboard AI capabilities is security, as highlighted in [46–48] . Drone communications are susceptible to cyber-attacks, making it crucial to protect the data being transmitted between the UAV and the cloud. Implementing security measures such as encryption and authentication protocols can protect the system from unauthorized access and data breaches. Additionally, implementing physical security measures such as tamper-proofing the onboard AI hardware can prevent malicious actors from tampering with the system. These security measures must be implemented at every stage of the system development and deployment to ensure the safety and privacy of data collected by UAVs. Nevertheless, these measures can affect the system's inference speed in a way that still has to be investigated.

Author Contributions: Conceptualization, A.K. and A.A.; methodology, A.K., A.A. and M.A.; software, Y.A. and A.A.; validation, A.K., A.A. and M.A.; formal analysis, A.K., A.A. and M.A.; investigation, A.K., A.A. and M.A.; resources, A.K.; data curation, A.A. and Y.A.; writing—original draft preparation, A.K., A.A., M.A. and L.G.; writing—review and editing, A.K., A.A., M.A. and L.G.; visualization, A.A. and M.A.; supervision, A.K., A.A. and M.A.; project administration, A.K.; funding acquisition, A.K. All authors have read and agreed to the published version of the manuscript.

Funding: The APC for this article was funded by Prince Sultan University.

Acknowledgments: We thank Prince Sultan University for facilitating the experiments on the university campus and financially supporting publication expenses.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results

References

1. Zanelli, E.; Bödecke, H. *Global Drone Market Report 2022–2030*; Technical report; Drone Industry Insights: Hamburg, Germany, 2022.
2. Ammar, A.; Koubaa, A.; Benjdira, B. Deep-Learning-Based Automated Palm Tree Counting and Geolocation in Large Farms from Aerial Geotagged Images. *Agronomy* **2021**, *11*, 1458. [CrossRef]
3. Gallego, V.; Rossi, M.; Brunelli, D. Unmanned aerial gas leakage localization and mapping using microdrones. In Proceedings of the 2015 IEEE Sensors Applications Symposium (SAS), Zadar, Croatia, 13–15 April 2015 ; pp. 1–6. [CrossRef]
4. Abdelkader, M.; Shaqura, M.; Claudel, C.G.; Gueaieb, W. A UAV based system for real time flash flood monitoring in desert environments using Lagrangian microsensors. In Proceedings of the 2013 International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GA, USA, 28–31 May 2013; pp. 25–34. [CrossRef]
5. Abdelkader, M.; Shaqura, M.; Ghommem, M.; Collier, N.; Calo, V.; Claudel, C. Optimal multi-agent path planning for fast inverse modeling in UAV-based flood sensing applications. In Proceedings of the 2014 International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, FL, USA, 27–30 May 2014; pp. 64–71. [CrossRef]
6. Benjdira, B.; Bazi, Y.; Koubaa, A.; Ouni, K. Unsupervised Domain Adaptation Using Generative Adversarial Networks for Semantic Segmentation of Aerial Images. *Remote Sens.* **2019**, *11*, 1369. [CrossRef]
7. Benjdira, B.; Ammar, A.; Koubaa, A.; Ouni, K. Data-Efficient Domain Adaptation for Semantic Segmentation of Aerial Imagery Using Generative Adversarial Networks. *Appl. Sci.* **2020**, *10*, 1092. [CrossRef]
8. Gulf News, Saudi Arabia: 131 People Went Missing in Desert Last Year. 2021. Available online: https://gulfnews.com/world/gulf/saudi/saudi-arabia-131-people-went-missing-in-desert-last-year-1.78403752 (accessed on 1 March 2023).
9. Kobaa, A. System and Method for Service Oriented Cloud Based Management of Internet-of-Drones. U.S. Patent US11473913B2, 15 October 2022.
10. Fortune Buisness Insights, Drone Surveillance Market. 2022. Available online: https://www.fortunebusinessinsights.com/industry-reports/drone-surveillance-market-100511 (accessed on 1 March 2023).

11. Ammar, A.; Koubaa, A.; Ahmed, M.; Saad, A.; Benjdira, B. Vehicle detection from aerial images using deep learning: A comparative study. *Electronics* **2021**, *10*, 820. [CrossRef]
12. Yeom, S.; Cho, I.J. Detection and tracking of moving pedestrians with a small unmanned aerial vehicle. *Appl. Sci.* **2019**, *9*, 3359. [CrossRef]
13. Ding, J.; Zhang, J.; Zhan, Z.; Tang, X.; Wang, X. A Precision Efficient Method for Collapsed Building Detection in Post-Earthquake UAV Images Based on the Improved NMS Algorithm and Faster R-CNN. *Remote Sens.* **2022**, *14*, 663. [CrossRef]
14. Koubaa, A.; Ammar, A.; Alahdab, M.; Kanhouch, A.; Azar, A.T. DeepBrain: Experimental Evaluation of Cloud-Based Computation Offloading and Edge Computing in the Internet-of-Drones for Deep Learning Applications. *Sensors* **2020**, *20*, 5240. [CrossRef] [PubMed]
15. Hossain, S.; Lee, D.j. Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU-based embedded devices. *Sensors* **2019**, *19*, 3371. [CrossRef]
16. Queralta, J.P.; Raitoharju, J.; Gia, T.N.; Passalis, N.; Westerlund, T. Autosos: Towards multi-uav systems supporting maritime search and rescue with lightweight ai and edge computing. *arXiv* **2020**, arXiv:2005.03409.
17. Vasilopoulos, E.; Vosinakis, G.; Krommyda, M.; Karagiannidis, L.; Ouzounoglou, E.; Amditis, A. A Comparative Study of Autonomous Object Detection Algorithms in the Maritime Environment Using a UAV Platform. *Computation* **2022**, *10*, 42. [CrossRef]
18. Pajares, G. Overview and Current Status of Remote Sensing Applications Based on Unmanned Aerial Vehicles (UAVs). *Photogramm. Eng. Remote Sens.* **2015**, *81*, 281–330. [CrossRef]
19. Nex, F.; Remondino, F. UAV for 3D mapping applications: A review. *Appl. Geomat.* **2014**, *6*, 1–15. [CrossRef]
20. Bhardwaj, A.; Sam, L.; Martín-Torres, F.J.; Kumar, R. UAVs as remote sensing platform in glaciology: Present applications and future prospects. *Remote Sens. Environ.* **2016**, *175*, 196–204. [CrossRef]
21. Torresan, C.; Berton, A.; Carotenuto, F.; Di Gennaro, S.F.; Gioli, B.; Matese, A.; Miglietta, F.; Vagnoli, C.; Zaldei, A.; Wallace, L. Forestry applications of UAVs in Europe: A review. *Int. J. Remote Sens.* **2017**, *38*, 2427–2447. [CrossRef]
22. Yao, H.; Qin, R.; Chen, X. Unmanned Aerial Vehicle for Remote Sensing Applications—A Review. *Remote Sens.* **2019**, *11*, 1443–1464. [CrossRef]
23. Messous, M.A.; Hellwagner, H.; Senouci, S.M.; Emini, D.; Schnieders, D. Edge computing for visual navigation and mapping in a UAV network. In Proceedings of the ICC 2020–2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6.
24. Liu, Q.; Shi, L.; Sun, L.; Li, J.; Ding, M.; Shu, F. Path Planning for UAV-Mounted Mobile Edge Computing with Deep Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2020**, *69*, 5723–5728. [CrossRef]
25. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602.
26. Afifi, G.; Gadallah, Y. Cellular Network-Supported Machine Learning Techniques for Autonomous UAV Trajectory Planning. *IEEE Access* **2022**, *10*, 131996–132011. [CrossRef]
27. Xia, W.; Zhu, Y.; De Simone, L.; Dagiuklas, T.; Wong, K.K.; Zheng, G. Multiagent Collaborative Learning for UAV Enabled Wireless Networks. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 2630–2642. [CrossRef]
28. Li, B.; Liu, Y.; Tan, L.; Pan, H.; Zhang, Y. Digital twin assisted task offloading for aerial edge computing and networks. *IEEE Trans. Veh. Technol.* **2022**, *71*, 10863–10877. [CrossRef]
29. Li, K.; Ni, W.; Yuan, X.; Noor, A.; Jamalipour, A. Deep Graph-based Reinforcement Learning for Joint Cruise Control and Task Offloading for Aerial Edge Internet-of-Things (EdgeIoT). *IEEE Internet Things J.* **2022**, *9*, 21676–21686. [CrossRef]
30. Qian, Y.; Sheng, K.; Ma, C.; Li, J.; Ding, M.; Hassan, M. Path Planning for the Dynamic UAV-Aided Wireless Systems Using Monte Carlo Tree Search. *IEEE Trans. Veh. Technol.* **2022**, *71*, 6716–6721. [CrossRef]
31. Wang, Y.; Chen, W.; Luan, T.H.; Su, Z.; Xu, Q.; Li, R.; Chen, N. Task Offloading for Post-Disaster Rescue in Unmanned Aerial Vehicles Networks. *IEEE/ACM Trans. Netw.* **2022**, *30*, 1525–1539. [CrossRef]
32. Yang, Z.; Bi, S.; Zhang, Y.J.A. Online Trajectory and Resource Optimization for Stochastic UAV-Enabled MEC Systems. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 5629–5643. [CrossRef]
33. Lyu, L.; Zeng, F.; Xiao, Z.; Zhang, C.; Jiang, H.; Havyarimana, V. Computation Bits Maximization in UAV-Enabled Mobile-Edge Computing System. *IEEE Internet Things J.* **2022**, *9*, 10640–10651. [CrossRef]
34. Hamasha, M.; Rumbe, G. Determining optimal policy for emergency department using Markov decision process. *World J. Eng.* **2017**, *14*, 467–472. [CrossRef]
35. El-Shafai, W.; El-Hag, N.A.; Sedik, A.; Elbanby, G.; Abd El-Samie, F.E.; Soliman, N.F.; AlEisa, H.N.; Abdel Samea, M.E. An Efficient Medical Image Deep Fusion Model Based on Convolutional Neural Networks. *Comput. Mater. Contin.* **2023**, *74*, 2905–2925. [CrossRef]
36. Sabry, E.S.; Elagooz, S.; El-Samie, F.E.A.; El-Shafai, W.; El-Bahnasawy, N.A.; El-Banby, G.; Soliman, N.F.; Sengan, S.; Ramadan, R.A. Sketch-Based Retrieval Approach Using Artificial Intelligence Algorithms for Deep Vision Feature Extraction. *Axioms* **2022**, *11*, 663–698. [CrossRef]
37. Meier, L.; Honegger, D.; Pollefeys, M. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 6235–6240. [CrossRef]

38. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv* **2022**, arXiv:2207.02696. https://doi.org/10.48550/ARXIV.2207.02696.
39. Wojke, N.; Bewley, A.; Paulus, D. Simple online and realtime tracking with a deep association metric. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 3645–3649.
40. Shafi, O.; Rai, C.; Sen, R.; Ananthanarayanan, G. Demystifying TensorRT: Characterizing Neural Network Inference Engine on Nvidia Edge Devices. In Proceedings of the 2021 IEEE International Symposium on Workload Characterization (IISWC), Storrs, CT, USA, 7–9 November 2021; pp. 226–237. [CrossRef]
41. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
42. Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; Upcroft, B. Simple online and realtime tracking. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AN, USA, 25–28 September 2016; pp. 3464–3468.
43. Ammar, A.; Koubaa, A.; Boulila, W.; Benjdira, B.; Alhabashi, Y. A Multi-Stage Deep-Learning-Based Vehicle and License Plate Recognition System with Real-Time Edge Inference. *Sensors* **2023**, *23*, 2120. [CrossRef]
44. Koubaa, A.; Ammar, A.; Kanhouch, A.; AlHabashi, Y. Cloud Versus Edge Deployment Strategies of Real-Time Face Recognition Inference. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 143–160. [CrossRef]
45. Zhu, P.; Wen, L.; Du, D.; Bian, X.; Fan, H.; Hu, Q.; Ling, H. Detection and Tracking Meet Drones Challenge. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 7380–7399. [CrossRef]
46. Krichen, M.; Adoni, W.Y.H.; Mihoub, A.; Alzahrani, M.Y.; Nahhal, T. Security Challenges for Drone Communications: Possible Threats, Attacks and Countermeasures. In Proceedings of the 2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH), Riyadh, Saudi Arabia, 22–24 May 2022; pp. 184–189.
47. Ko, Y.; Kim, J.; Duguma, D.G.; Astillo, P.V.; You, I.; Pau, G. Drone secure communication protocol for future sensitive applications in military zone. *Sensors* **2021**, *21*, 2057. [CrossRef]
48. Khan, N.A.; Jhanjhi, N.Z.; Brohi, S.N.; Nayyar, A. Emerging use of UAV's: Secure communication protocol issues and challenges. In *Drones in Smart-Cities*; Elsevier: Amsterdam, The Netherlands, 2020; pp. 37–55.