Dynamic Computation Off-loading and Control based on Occlusion Detection in Drone Video Analytics

Rajeswara Rao Ramisetty*
Jawaharlal Nehru Technological University-Kakinada
Vizianagaram, India
raob4u@jntukucev.ac.in

Chengyi Qu*
University of Missouri-Columbia
Columbia, Missouri
cqy78@mail.missouri.edu

Rumana Aktar*
University of Missouri-Columbia
Columbia, Missouri
rayy7@mail.missouri.edu

Songjie Wang
University of Missouri-Columbia
Columbia, Missouri
wangso@missouri.edu

Prasad Calyam
University of Missouri-Columbia
Columbia, Missouri
calyamp@missouri.edu

Kannappan Palaniappan
University of Missouri-Columbia
Columbia, Missouri
pal@missouri.edu

ABSTRACT

Unmanned Aerial Vehicles (UAVs) or drones equipped with cameras are extensively used in different scenarios such as surveillance of hazardous locations, disaster response and crime fighting. The related video streaming/analytics requires real-time drone-to-Ground Control Station (GCS) communication and computation co-ordination for desired user Quality of Experience (QoE). In situations where the quality of the video can be affected by occlusions (e.g., image distortion, frame stalling) due to network bottlenecks, there is a need to dynamically make decisions on the computation offloading and networking protocols in order to properly handle the video data for real world application purposes. In this paper, we propose a novel function-centric computing approach that helps a user to perform drone video analytics to assess a wide-area scene to chart a plan of action. Our approach involves handling network impairments affecting the switching between high resolution/low resolution video capture, or change of camera direction for assessment of the scene effectively. It also features a novel video quality enhancing algorithm based on occlusion-detection that adapts to video impairments related to image distortion and frame stalling. Our experiment results from a realistic testbed show that our approach can efficiently choose the suitable networking protocols (i.e., TCP/HTTP, UDP/RTP, QUIC) and orchestrate both the camera control on the drone, and the computation off-loading of the video analytics over limited edge computing resources. The performance improvements for computation off-loading involving our video quality enhancing algorithm are shown for different network conditions in terms of occlusion rate and processing times.

*These authors contributed equally to this research.

ICDCN 2020, January 4–7, 2020, Kolkata, India

ACM ISBN 978-1-4503-7751-5/20/01…$15.00
https://doi.org/10.1145/3369740.3369793

CCS CONCEPTS

• **Networks** → **Transport protocols**; • **Information systems** → *Users and interactive retrieval.*

KEYWORDS

Drone video analytics, Multimedia networking protocols, Computation off-loading, Occlusion detection, Network management

ACM Reference Format:
Rajeswara Rao Ramisetty, Chengyi Qu, Rumana Aktar, Songjie Wang, Prasad Calyam, and Kannappan Palaniappan. 2020. Dynamic Computation Off-loading and Control based on Occlusion Detection in Drone Video Analytics. In *21st International Conference on Distributed Computing and Networking (ICDCN 2020), January 4–7, 2020, Kolkata, India.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3369740.3369793

1 INTRODUCTION

In the last few years, Unmanned Aerial Vehicles (UAVs), also known as drones, have been extensively used in different scenarios in urban and rural area control such as disaster response, surveillance of smart city, crime fighting and smart farming [18]. Most commercially used drones are equipped with high-resolution cameras that are able to visualize and monitor target status, e.g., object recognition, counting and tracking purposes.

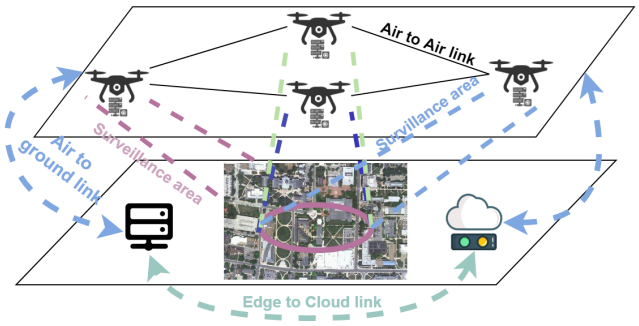


Figure 1: Overview of multiple drone communication and GCS computation using a FANET and an edge computing infrastructure.

State-of-the-art video analytics processing applications are increasing using drone video farming that requires high-performance

computing resources and real-time communication. The problems of insufficient computing resources and edge networking speeds can be addressed using computation off-loading to a Ground Control Station (GCS) [14]. However, computation off-loading cannot always be used for tactical drones computation and control due to the dynamic wireless channel and energy consumption constraints on the drone-edge computing for processing high-resolution images. In edge networks, a variety of environmental conditions may affect the performance of the video streaming between the drone and GCS. This in turn affects the performance of video streaming in terms of end-to-end delay, frame blurring, stalling and distortion. Based on our literature survey and important observations from [13], there is a clear lack of mechanisms to efficiently coordinate the networking protocols in conjunction with drone control orchestration and computation off-loading during drone video analytics.

As shown in Figure 1, we consider an application scenario comprising of a network of drones based on the concepts borrowed from prior work on Flying Ad hoc Networks (FANETs) [1]. A multi-UAV system can operate in a centralized or decentralized manner. In a decentralized system, the UAVs need to explicitly cooperate on different levels to achieve the system goals and exchange information to share tasks and make collective decisions. We assume that the drones are cooperatively working in a decentralized way when recording the video of a scene at different angle. Even if the connection among one of the drones to the GCS is interrupted or disconnected, we suppose that the inter-drone communication may still be possible as outlined in [11].

In this paper, we present a novel *dynamic computation off-loading and control scheme* that detects occlusions that impair user Quality of Experience (QoE), and coordinates intelligent processing in drone video analytics. The intelligent processing considers the trade-off in processing time vs. tracking/accuracy rate using a Function Centric Computing (FCC) architecture [6]. The FCC architecture utilizes the drone edge, GCS and cloud computing resources and decomposes an application over these resources in the form of microservice functions/tasks. We summarize the major novel contributions of this paper as follows:

- We describe a drone video analytics application that supports FCC by decomposing a pipeline for video surveillance in the form of microservice functions/tasks that are deployed over a mixture of drone/GCS/cloud computing resources communicating via RESTful web services [17].
- We also present a novel dynamic computation off-loading and control scheme based on occlusion detection in drone video analytics. Our approach supports FCC to optimally trade-off processing time vs. tracking/accuracy rate among the different computing architectures, i.e., drone, GCS and cloud. The goal of our approach is to dynamically change networking protocols (i.e., TCP/HTTP, UDP/RTP, QUIC), and handle network impairments affecting the switching between high resolution/low resolution video capture, or change of camera direction for assessment of the scene effectively.
- Lastly, we deploy a drone/GCS/cloud testbed in the GENI cloud infrastructure [16] and span it in multiple geographical locations using novel metrics such as occlusion rate and processing times. We also evaluated our scheme under various network conditions and realistic application settings.

The rest of the paper is organized as follows: Section 2 presents related work. In Section 3, we formulate our problem with occlusion-based settings, categorize the occlusion formats with transport layer protocol considerations, and introduce our drone video analytics application pipeline. In Section 4, we detail our novel dynamic computation off-loading scheme for real-time drone video analytics. Section 5 describes our experimental testbed settings, performance metrics and evaluation results. Section 6 concludes the paper.

2 RELATED WORK

Computation off-loading on drones. Real-time video analytics applications pose new challenges in processing intensive media-rich data on the drone due to its limited computing power. It is difficult to meet user QoE [3] expectations when low-latency and highly accurate video analytics is required to be performed [25].

In [6], a novel policy-based function-centric computation off-loading (FCC) scheme was presented for off-loading decision making. The decision to off-load either to an edge, cloud or FCC for real-time video analytics is done by considering the performance vs. cost factors as evaluation parameters. However, this prior work did not consider factors such as network connection quality and protocol selection needed on the drone during computation and data transfer. In [27], the authors proposed an off-loading scheme for a drone cluster with high computing tasks to borrow the computing resources from nearby clusters that have low computing tasks. Due to the constrained drone resources (i.e computation power, energy, human operations, etc.,) the off-loading decision is made statically based on the available computational resources of the neighbor clusters. In both of these works, the decision making process is not dynamic and cannot adapt to environmental changes such as network connection bottlenecks and austere edge computing.

In [13], the authors proposed an adaptive computation off-loading drone system architecture (ACODS) with reliable communication to preserve energy consumption on the drone. A prediction module was used to estimate the overall system response time to decide if the computation needs to be performed on the drone or off-loaded to a GCS. This work, however, does not consider partial off-load of computation workloads to maximize performance and to reduce the the system delays caused by data transmission during drone-to-GCS communication. Our work overcomes this issue by introducing the FCC concept that partially off-loads computation workload from the drones, while simultaneously performing a reasonable level of computation on the drone device itself.

Authors in [10] present a framework for UAV-based video monitoring with a FANET. The FANET consists of UAVs with multi-access edge computing (MEC), which helps reduce response delays by providing higher network bandwidth and provides off-loading capabilities to reduce computation workload. However, in this work, the increase in the performance of the whole aerial-ground MEC platform and an off-loading decision-making scheme is not presented. Specifically, we propose a dynamic off-loading scheme that efficiently off-loads the computation among edge, cloud or FCC architecture based on thresholds of occlusion detection.

Dynamic Decision Making on computation off-loading. Due to the change in environments during UAV's movement, the computation off-loading decision making needs to be dynamically made in order to keep system performance at a high level and to provide satisfactory user QoE [5].

An energy-aware dynamic computation off-loading scheme for object recognition and tracking is proposed in [26]. The proposed off-loading scheme considers both the dwell time of the moving target object and the network failure rate to estimate the system response time for recognizing and tracking of a moving object. A mobile-cloudlet-cloud architecture (MOCHAA) is presented in [24], in which face recognition applications were used to evaluate the ability of the system to minimize the response time in airport security surveillance. The system architecture deals with how to divide computation loads among the servers in order to get minimal response. However, it does not consider the mobility of the edge device which could affect the network connection quality and thus the transmission delay. A real-time object recognition system using a camera equipped mobile device is proposed by [29]. In this system, the algorithms for object recognition and labeling are off-loaded to the cloud due to its significant computation overhead. Also, off-loading to a cloud server is determined by energy consumption and computation delays of the system. However, it does not consider the variability in network conditions between mobile devices and a cloud server. A solution using deep reinforcement learning is proposed to dynamically off-load computation as part of a sequential decision making for cloud robotics [7]. In this work, vision deep neural networks (DNN) were used for decision making in off-loading computation workloads. This approach has shown improved performance during off-loading, thus allowing robots to significantly transcend their on-board sensing accuracy. However, the off-loading scheme is limited at a device level or at individual cloud servers. In contrast, our approach proposes dynamic off-loading scheme among edge, cloud and FCC, providing more available choices and flexibility to the application users.

Occlusion-awareness on Aerial Video Analytics. The overhead imagery recognition poses many challenging tasks than a fixed ground level camera because of low resolution or high resolution imagery patterns [2]. Here a major challenging task involves handling the large intra-class variation in activities including variations in resolution scale, target (e.g., visual appearance, speed of motion) and environment (e.g., lighting condition, occlusion) [28]. The meaningful structures in video are extracted through the unsupervised learning of temporal clusters and are associated with the metadata [8]. A flexible dual TCP-UDP streaming protocol (FDSP) is used for high-quality video streaming [9]. Here, this FDSP approach results in lower packet loss compared to UDP-based streaming. The performance of QUIC vs. TCP for a better QoE in video streaming is studied in [23]. Our approach involves handling network impairments affecting the switching between high resolution/low resolution video capture, or change of camera direction for assessment of the scene effectively. It features a novel video quality enhancing algorithm based on occlusion-detection that adapts to video impairments related to image distortion and frame stalling.

Function-centric Computing. In [4], Service Function Chaining (SFC) was proposed to compose and maintain the services in a geo-distributed cloud infrastructure. Each service is considered as a function and the task is represented as a service function chain. The compute / network resources are allocated to each service in a cloud infrastructure in SFC. Studies have shown that SFC orchestraction deployed on the realistic edge / cloud has given good performance. This Service Function Chain (SFC) was first used by [6] in policy based FCC for off-loading. In this work, the given application is decomposed into functions, such as microservices, and each functions can be off-loaded to different sites for execution. The proposed FCC based off-loading approach has given satisfactory performance with respect to frames per-second vs. a cost factor for a given drone-based video analytics application. In our proposed approach, we extend the FCC approach with dynamic decision making in computation off-loading with consideration of device mobility, system response time, network quality and energy consumption for processing streamed video from UAVs.

3 OCCLUSION-BASED PROBLEM FORMULATION

In this section, we first introduce our problem formulation, which includes occlusion type and rate, occlusion influencing factors, and how these occlusion types and influencing factors affect computation off-loading decision making. We then present our function chain for processing of streamed video, and how we decouple the function chain into FCC functions, as described in [6].

3.1 Drone operations influence on occlusion type and rate

In our proposed scenario, a fleet of drones is embedded with high-resolution cameras and one control drone is in charge of controlling the drone swarm. We divide our drone control into several different control panels, including navigation control, platform control and sensor/camera control. It is needed in advance and the guide includes changing the location or the speed of the drone fleet. Platform control mainly focuses on changing the position, pose or height of the drone swarm to clearly regain the vision of any blocked objects. This decision could be made by the control drone itself, edge server on the GCS, or human operators on the ground. Sensor/camera control is responsible for changing the camera information, such as zooming-in/zooming-out on Pan-Tilt-Zoom (PTZ) cameras or reset of the sensor e.g., switching to infrared or night-vision sensors, to avoid missing targets.

To make sure the drone fleet control decision is accurate and on-time, a standard occlusion detection and occlusion rate calculating algorithm are provided. In our approach, we consider most of the data transmission through wireless communication between drones and GCS, and most of the video streaming uses network layer protocols to deal with network transport issues. We observed the spatial and temporal occlusions that occur during video streaming of various types of file formats and codec types with different networking protocols. Table 1 summarizes our experiments with TCP/QUIC [23], UDP [9]/RTP [22] networking protocols that occurred during video streaming using MPEG/MPEG.1 or MP4/H.264 codecs. Figure 2 illustrates the broadly different spatial and temporal occlusions that were recorded in the video frames in our experiments.

Table 1: Comparison of occlusion types found for different video formats, codec types and networking protocols in experiments with the 5 Mbps available network bandwidth setting.

Original Resolution	File Format	Codec Type	Protocol	Lossy/Lossless	Impairment Type
1904*1071	MP4	H.264	UDP/RTP	Lossy/Lossless	Spatial Distortion
	MP4	H.264	TCP/QUIC	Lossy/Lossless	Temporal Stalling
1344*756	MP4	MP4	UDP/RTP	Lossy	Spatial distortion
	MP4	MP4	TCP/QUIC	Lossy	Temporal Stalling
1360*765	MPEG	H.264	UDP/RTP	Lossy	Distortion/frame overlap/green lines
	MPEG	H.264	TCP/QUIC	Lossy	Stalling
1920*1080	MPEG	MPEG-1	UDP/RTP	Lossy	Distortion
	MPEG	MPEG-1	TCP/QUIC	Lossy	Stalling

Algorithm 1: Spatial Impairment

Input: $inFrame, GT$
Result: $sptlImpairment$

$[M, N] \leftarrow sizeOfImage(inFrame(1))$;
$n \leftarrow getFrameCount(inFrame)$;
$C \leftarrow 36$;
for i=1:n **do**
  $imageDiff(i) \leftarrow abs(GT(i) - inFrame(i))$;
  $occludedPixel(i) \leftarrow nonzeroPixels(imageDiff(i))$;
  $occludedPct(i) \leftarrow \frac{occludedPixel(i)*100}{M*N}$;
  $intensityDiff(i) \leftarrow \frac{imageDiff(i)}{M*N}$
end
$sptlImpairment \leftarrow C * mean(occludedPct * intensityDiff)$

(a) Video frames with spatial distorting occlusions

(b) Video frames with temporal stalling occlusions

Figure 2: Illustration of spatial and temporal occlusions that are caused by low available network bandwidth situations during MPEG/MPEG-1 or MP4/H.264 video streaming using these networking protocols: (a) UDP/RTP and (b) TCP/QUIC.

Based on our observations of stalling and blurring i.e., the two major occlusion types occurring during video streaming, we calculate the occlusion rate as shown in Algorithm 1.

Spatial Impairment Rate: Spatial impairment in a frame can be defined as how dissimilar the received content after transmission is from the original content captured by the drone or an edge computing device. Given the ground truth data, the amount of spatial impairment present in each frame can be evaluated simply comparing the frames before and after transmission as detailed in the steps shown in Algorithm 1.

Temporal Impairment Rate: To get the sense of frame stalling or temporal impairment from a video streaming session, we compare frame at time i, with frame at time $i-1$. If both are same, then there is a temporal impairment. We calculate the total number of stalled or buffered frames in a video sequence. Temporal impairment rate is defined as the ratio of number of buffered frames and total number of frames in a video sequence.

3.2 Function Chain for Video Processing Application

Our drone video processing application performs objects motion detection for aerial images and their subsequent classification. An example result of this analytics application is show in Figure 3. To implement the FCC pipeline, we decompose our video analysis application and the occlusion calculation algorithm into separate microservices. Streamed video data is collected after transmission via different protocols, including TCP, UDP, RTP and QUIC. After the video is captured, the video processing functions (microservices) resides within Docker containers that can be quickly deployed on appropriate nodes after computation off-loading decision making. The decoupled microservices from the application allow us to independently execute specific functions at desired location to evaluate our different computation off-loading schemes, i.e., edge, cloud, and FCC schemes.

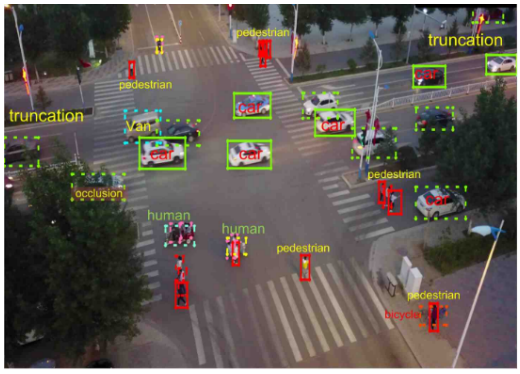


Figure 3: Example result of object classification on a surveillance video from one of the drones in the drone fleet.

The implemented drone video analytics pipeline is show in Figure 4, and has the following functions implemented as RESTful web service interfaces:

The pre-processing endpoint. This endpoint (*preProcessing*) is responsible for extracting the sections of an image that changed between succeeding frames in a video using classic computer vision techniques, see Section 5.1. This allows it to not only count object whose positions have changed from preceding frames, but also allows for labeling them accordingly. This endpoint includes gray scaling function, blurring function and motion detection function.

The occlusion detection endpoint. This endpoint includes an occlusion detection function. The (*occlusionDetection*) is responsible for detecting the occlusion of the streaming video after transmission and calculating the occlusion rate. The main function contains two modules i.e., occlusion categorization and occlusion rate calculation. In our approach, we only consider two types of occlusion i.e., blurring rate and stalling rate, and use the same algorithm settings to calculate both.

The classifier endpoint. This endpoint includes motion classifier and moving object counter function. This endpoint (*objectClassifier*) features a YOLOv3 classifier running on TensorFlow that was pre-trained on COCO [21]. It receives a frame from an image as input, and is able to quickly carry out object recognition on it and return bounding boxes and labels for all the objects occurring in the frame. In addition, this endpoint is also responsible for processing counting methods and for returning results on moving objects with the results from a pre-processing endpoint.

The function chain definition endpoint. This endpoint (*Next Server*) is used to dynamically determine which server is called to execute the next function. This allows programmatic access to define where the classifier is executed. The same Docker image is supported by all nodes, including controller drone, GCS with edge server, and a cloud server.

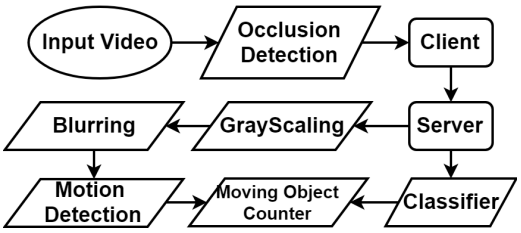


Figure 4: Image processing pipeline for video analytics.

4 DYNAMIC COMPUTATION OFF-LOADING DECISION MAKING SCHEME

In this section, we introduce our novel dynamic computation off-loading decision making scheme. This scheme allows us to not only facilitate trade-offs in performance vs. processing speed of real-time drone video analytics, but also aids in making decisions in selecting edge, cloud or FCC for data processing. We also consider network failure and recovery time in the drone video transmission. For real-time processing and the high-quality of video analysis, we calculate a occlusion rate metric during drone flying and transmission guidance.

4.1 Network delays in dynamic decision making

In this section, we characterize the impact of network failure on a drone-based surveillance system. In our proposed scheme, drone fleet and GCS are connected with a wireless edge network, which is intermittently unstable and experiences frequent network failure events. In other words, the computation off-loading will have a significant problem with unexpected network failure events that can be caused by e.g., mobility of the drone, weather conditions and occlusion caused by low-resolution video capture. To address this problem, the network recovery time ζ_α is estimated before we make computation off-loading decisions on the application pipeline tasks. If the recovery time ζ is less than the network recovery time ζ_α, then FCC is adopted for the given application. This dynamic off-loading decision based on network condition for FCC enables us to reduce the overall processing time. If there is no network failure, then the off-loading is directed to any available high-performance edge computing resources.

The network recovery time ζ_α is calculated as follows: Let the network error rate be λ_i at time i. For a given network, if we define the response time from the drone to the GCS server as ζ, then $E[\zeta]$ follows the Poisson distribution [15]. If $E[\zeta]$ is defined as the response time for the network without any failure, then it is identical to ζ. It is assumed that during a network failure event, the network will retry to obtain connectivity continuously after waiting for a recovery time q. Then $E[\zeta]$ is represented by three distinct terms p,q,$E[\zeta]$, respectively; where p is defined as the time interval between the point of initiation of the task and the network failure event occurrence.

$$\zeta_\alpha = E[\zeta] = \begin{cases} p + q + E[\zeta] & \text{if } p < q \\ \zeta & otherwise \end{cases} \tag{1}$$

By the law of total expectation $E[\zeta] = \int_0^\zeta (p+q+E[\zeta]) f_p(p)dp + \int_\zeta^\infty \zeta f_p(p)dp$, $E[\zeta]$ can be represented by:

$$E[\zeta] = \frac{\int_0^\zeta (p+q-\zeta) f_p(p)dp + \zeta}{1 - \int_0^\zeta f_p(p)dp} \tag{2}$$

By substituting $\lambda_i e^{-\lambda_i x}$ for $f_p(p)$, where λ_i is always greater than 0, we obtain

$$E[\zeta] = \frac{\int_0^\zeta (p+q-\zeta)\lambda_i e^{-\lambda_i x} + \zeta}{1 - \int_0^\zeta \lambda_i e^{-\lambda_i x}} \tag{3}$$

During network recovery, drones with computation resources on-board can be controlled to be responsible for taking a part in the video analytics work. The processing could be executed in parallel by the fleet of drones if the communication links are capable of handling the underlying message passing. We suppose that some of the workload tasks (i.e., those tasks that cannot be decoupled) must be done on the edge servers with high-performance computing resources. In such cases, we buffer the video data instead of pursing off-loading. Thus, our proposed scheme can handle network failure events without compromising the accuracy as well as tracking performance, as opposed to performing the computations using the drone-only or edge-GCS computation off-loading options.

4.2 Policy-based Function-Centric Computing in dynamic decision making

In our recent prior work [6], we assume that the both the drone and GCS edge computation resources will be leveraged to perform tasks such as pre-processing in the video analytics pipeline. Hence, the decision of scheduling a task/function to be executed at a suitable location is an optimization problem. For the pre-condition, we assumed that the computation resources on-board a drone is less in terms of computation power than at a GCS that is co-located with a high-performance computing edge server. If the GCS has access to high-performace computation resources, we assume that the resources are abundant and real-time video processing can be achieved without waiting time. In the following, we first describe our supported policies, and then detail our occlusion-based computation off-loading algorithm that uses FCC.

The polices that we consider include: (i) the *real-time control* policy, which is required during the video streaming. A dynamic control strategy must be determined for the drone fleet and the GCS in real-time based on the response time with low-latency; (ii) the *FCC availability* policy ensures that a video analytics application is function-based or microservices-based, and the individual functions can be executed at different locations; and (iii) the *video performance* policy, which calculates the occlusion rate of the video during transmission as well as the processing speed. A few metrics that we use to implement this policy include: tracking rate, accuracy rate, and system response time; note that we describe these metrics later in detail in Section 5.

4.3 Occlusion-based dynamic decision making in drone control

The real-time control relies on estimating the quality of the video streamed by the drones. The level of occlusion is detected at the drone by estimating the peak signal-to-noise ratio (PSNR), as shown in Equation 5. We remark that PSNR is a widely used metric to estimate the quality of images that undergo degradation when being transmitted on a communication link. PSNR is estimated by using mean square error (MSE), and the image quality at the destination of a communication link can be estimated as:

$$MSE = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M * N} \tag{4}$$

where M, N define the size of the image, and m, n are pixels.

$$Threshold = PSNR = 20log_{10}\frac{V_{Peak}}{MSE} \tag{5}$$

Here we assume that impairments in the video can be rectified effectively if the PSNR is less than or equal to a fixed threshold (≤30%). If the PSNR value is within such limits, then the impairments in the video can be mitigated by instructing the drone to stream higher-resolution video through the drone controller as shown in Figure 4. If the PSNR threshold is (>30%), then the drone communicates to a nearby drone to capture the required high-quality video in the FANET [12]. This process iterates until the required high-resolution quality of the video is streamed as part of the video farming process.

Further, the drone control (as shown in Figure 5) checks if the quality of the streamed video from drone has a PSNR threshold of (≤30 %). In such cases, the impairments in the video can be rectified by the actions of camera control directed by navigation control to get the necessary high-resolution video streams in the video farming. Depending on the purpose that the drone is for, either of the above two actions are performed on the drone to satisfy user QoE expectations dynamically. For application pipelines involving object tracking or object detection, the drone can be instructed by the drone control module to shift to a low-resolution (to save on-board drone storage when high accuracy is not necessary for object recognition/counting) or a high-resolution (for increasing accuracy of object recognition/counting) setting.

4.4 Algorithm for dynamic computation off-loading and decision making

To detect occlusions and direct computation off-loading, the drone controller and GCS execute our proposed dynamic computation off-loading decision making as per the scheme shown in Figure 5 and the corresponding steps outlined in Algorithms 2 and 3.

Algorithm 2: D-FCC: Network Failure Module

Input: $Video$:= video data to analyze; $Drone$:= IP:Port; $Network_{status}$:=1 if Network failure, 0 Otherwise; $Recovery_{time}$:= t
Output: URI:= off-loading RESTful API URI; $Network\_strength\_function$, $Buffer\_on\_drone$:= raw data ready to off-load

```
1  begin
2      URI ← http://
3      Data ← Video
4      if Network_status == 1 then
5          if P.fcc_availability == 1 then
6              URI ← URI ∪ Drone/functionCentricProcessing
7              Data ← Data ∪ Recovery_time
8          end
9          else
10             Buffer_on_drone ← Data
11         end
12     end
13     else
14         Network_strength_function ← URI ∪ Data
15     end
16 end
```

We start by checking if a network failure event occurs at the beginning of the application (line 4 in Algorithm 2). If the network fails, and if the application can be decoupled into functions, we will execute part of the low computation resources needing functions on the drone and wait for network to be recovered (line 6, 7 in Algorithm 2). On the contrary, if the application is highly coupled, we will store the streaming video data into an on-board drone buffer to avoid packet/information loss (line 10 in Algorithm 2).

If the network health status is good for transmission, we will choose to use the edge high-performance computing resources for computation off-loading, if they are available. This in turn will result in better performance on tracking and accuracy in the object detection application pipeline. In this case, we will also choose off-loading to cloud resources (if they are available in order to increase the accuracy), instead of using low performance edge resources (line 6 in Algorithm 3).

In situations where there are no high-performance computing resources available at the edge, we choose to perform real-time control if we detect occlusions in the video frames. The occlusion-detection thus influences the subsequent real-time video farming and processing steps. Our scheme will provide guidance to the video

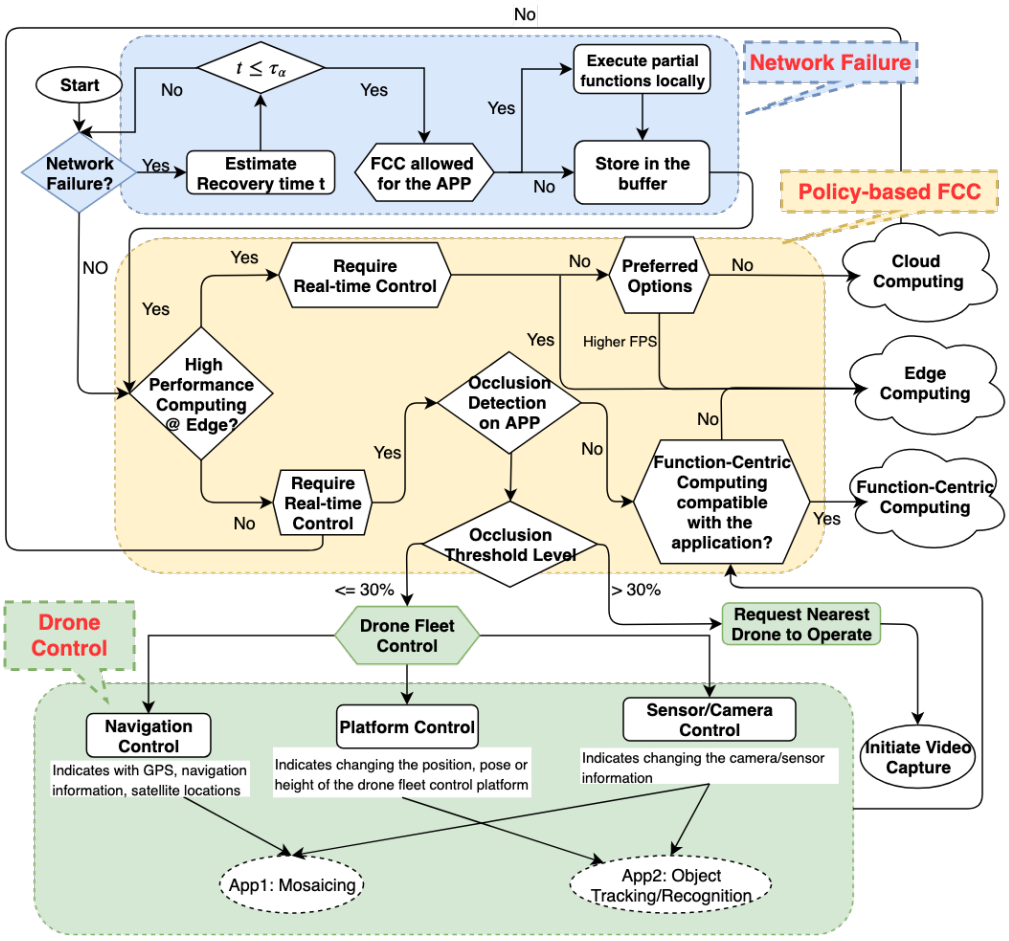


Figure 5: Illustration showing our dynamic computation off-loading and control policy based on occlusion detection using FCC off-loading scheme for FANET-based real-time drone video analytics.

capturing drone to change the drone location, speed, or change the resolution and networking protocol settings to guarantee the fluency and sharpness of the video streaming (line 19 in Algorithm 3). Further, we check whether the video analytics application is amenable to FCC (i.e., it can be decoupled into functions for placement of computation at different locations) (line 21 in Algorithm 3) to improve both its performance and the processing speed. Thus, by utilizing the high-performance computing resources intelligently during network failure events or cases of occlusion detection due to low-resolution video capture occurring in the drones, we can increase the tracking speed and accuracy in the object detection pipeline.

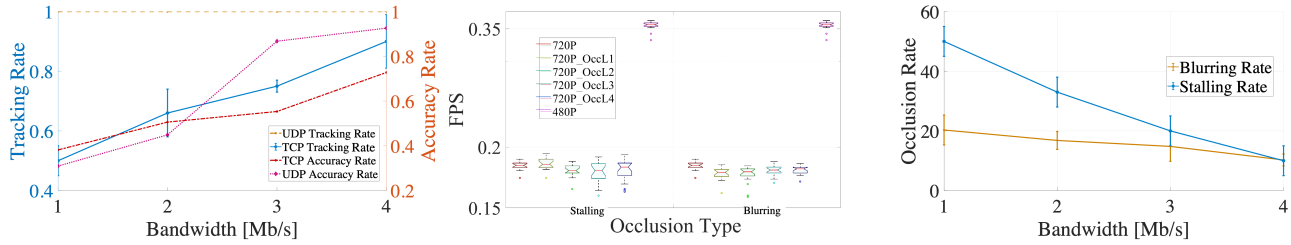
5 PERFORMANCE EVALUATION

In this section, we first describe the experimental setup. Following this, we discuss the evaluation results using metrics such as occlusion rate and stalling rate for different networking protocols under a variety of network health conditions considering different computation off-loading architectures for a drone video analytics pipeline involving object tracking and counting.

5.1 Experimental Setup

Our experiments consider several drone surveillance video streams from a VisDrone dataset [30] with different VGA resolutions (1080p: 1920 x 1080; 720p: 1280 x 720; 480p: 854 x 480; 360p: 640 x 360) in an analytics pipeline (see Figure 4) to track and count moving objects (e.g., cars, trucks and pedestrians). According to the VisDrone project, the videos are acquired by various drone platforms, i.e., DJI Mavic, Phantom series (3, 3A, 3SE, 3P, 4, 4A, 4P), including different scenarios across 14 different cites in China. The analytics application we use in our experiments is publicly available at [20]. As illustrated earlier in Figure 1, the farmed video data from a fleet of drones is captured and stored at different resolutions.

Our geo-distributed drone/GCS/cloud testbed setup includes 2 micro-processors, 1 edge server and 1 cloud server reserved in a GENI rack [16] at the Missouri InstaGENI site; 2 micro-processors and 1 HPC edge server. Micro-processors are NVIDIA Jetson Nano [19] with a configuration of 128-core Maxwell GPU, Quad-core ARM A57 CPU and 4G RAM. Edge server without HPC capabilities has 12 cores Intel Xeon CPU and 16 GB RAM and has the same settings on a cloud server at the Missouri InstaGENI site. Edge



(a) (b) (c)

Figure 6: Experiment results under different Network Bandwidth settings [in Mb/s] (1, 5, 10, and 15) with performance metrics impact on: (a) Tracking Rate and Accuracy Rate under 720p video resolution; (b) different level of occlusion and corresponding processing speed (FPS) with 2 standard video resolutions (480p and 720p); and (c) Blurring rate and Stalling rate under 720p video resolution.

Algorithm 3: D-FCC: Impairment detection and policy module

Input: $Video$:= video data to analyze; P:= set of policies; $Edge$:= IP:Port; $Cloud$:= IP:Port; $Drone$:= IP:Port; $Impairment_{rate}$:=ρ; $Impairment\_threshold$:=30%; $Edge_{HPC}$:= 1 if Edge HPC is available, 0 otherwise
Output: URI:= off-loading RESTful API URI; $Data$:= JSON data to off-load

begin
  $URI \leftarrow$ http://
  $Data \leftarrow Video$
  if $Edge_{HPC} == 1$ **then**
    if $P.real\_time == 0$ **then**
      $URI \leftarrow URI \cup Cloud$
    end
    else
      $URI \leftarrow URI \cup Edge$
    end
    $URI \leftarrow URI \cup$ /fullVideoProcessing
  end
  else
    if $P.real\_time == 0$ **then**
      //without real-time control policy
      $URI \leftarrow URI \cup Cloud \cup$ /fullVideoProcessing
    end
    else if $Impairment_{rate} <= Impairment\_threshold$ **then**
      //drone-control policy
      $Drone_{control} \leftarrow Drone_{control} \cup (P.resolution\_change \cap P.protocol\_change)$ **if** $P.fcc\_availability == 1$ **then**
        ///FCC avaliablity policy
        $URI \leftarrow URI \cup Edge \cup$ /functionCentricProcessing
        $Data \leftarrow Data \cup$ {"NextFunction" : $Cloud$}
      end
      else
        $URI \leftarrow URI \cup Edge \cup$ /fullVideoProcessing
      end
    end
    else
      $Request\_nearest\_drone \leftarrow Impairment_{rate} \cup Data$
    end
  end
end

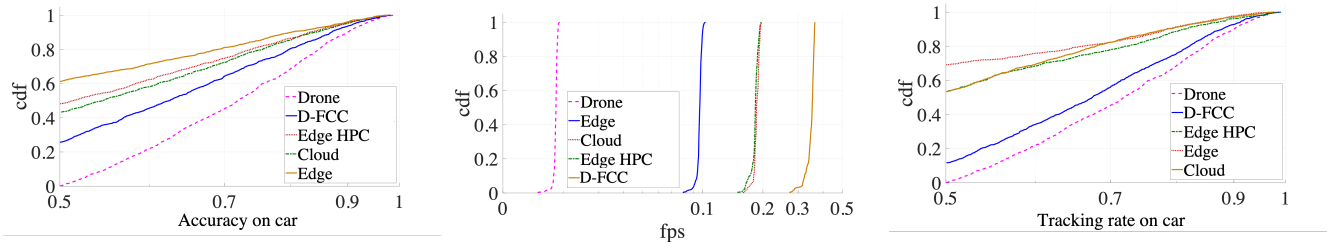
server with HPC settings is a NVIDIA Jetson AGX Xavier [19] that has a configuration of 8-Core ARM v8.2 64-bit NVIDIA Carmel CPU, 16GB RAM and 512-core GPU with 64 Tensor Cores. All the servers above support Docker containers and can execute a video analytics function without running an entire application pipeline.

We evaluated the performance of our video analytics by comparing results of 3 types of computation off-loading strategies and the computation on-boarding strategy. More specifically, we compared: cloud computing (*Cloud*), full off-loading (edge computing with/without HPC (*Edge*/*EdgeHPC*), Function-centric computing (*FCC*) and full on-boarding onto a drone compute edge (*Drone*).

5.2 Evaluation Results

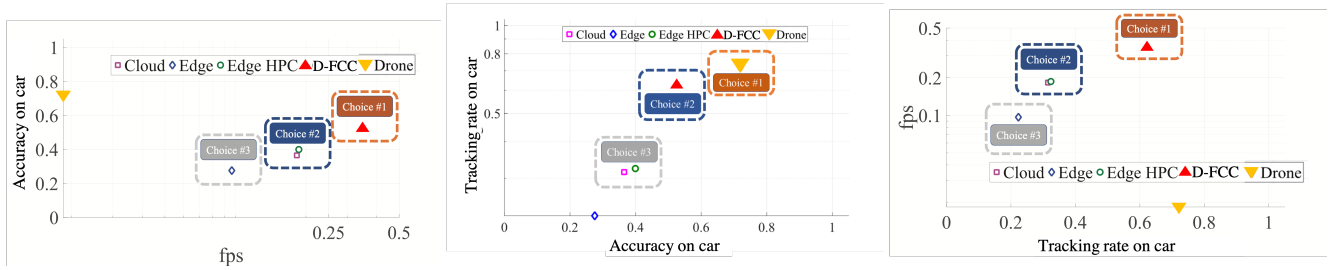
5.2.1 TCP and UDP Protocols Comparison. We conducted a series of experiments in our testbed to validate the effectiveness of our proposed *FCC* scheme by collecting the impairment rate and distortion rate for a given video stream captured by drones in a FANET setting. Figure 6 shows the performance trade-off between the TCP and UDP protocol configurations on the drones for the application pipeline under different conditions. The drone cameras resolution is fixed at 720 pixels. As shown in Figure 6a, we consider the tracking rate is always 100% for UDP for high-speed network bandwidth conditions. The performance of tracking rate is not satisfactory for the TCP protocol even for higher network bandwidth conditions. However, the tracking rate increases by increasing the available network bandwidth between the drones and the GCS. Moreover, we can observe from the experiments that choosing the UDP Protocol over TCP is more suited for applications such as object tracking. From Figure 6a, we can note for applications such as object recognition that the accuracy rate is always satisfactory for the video streamed by using the TCP protocol. However, for higher network bandwidth availability conditions, the accuracy rate is better when choosing the UDP Protocol. Moreover, for applications such as object recognition, choosing the TCP protocol at lower network bandwidth availability conditions is more suited than choosing the UDP protocol on the drone. The significant difference can be seen in the cases of limited network bandwidth availability conditions where we observe that the TCP protocol performance is better for fixed resolution cases (i.e., 720 pixels) than the UDP protocol performance.

5.2.2 Occlusion Rate Results. Figure 6b shows the performance results related to the different types of occlusions, and the corresponding processing times for various pixels ranges. Occlusion caused by video frame stalling occurs during the transmission of video by either TCP or QUIC networking protocols. Whereas occlusions caused by blurring occurs during the transmission of video by either UDP or RTP networking protocols. Our experiments use the original video sequences with 720 pixels and also the corresponding video sequences with varying pixels ranges. From the experiment results, we can conclude that the processing time increases with respect to increase in the pixel range for all type of networking protocols, and for all types of occlusions as well. However, video with 480 pixels processing time shows ideal performance in all



(a) (b) (c)

Figure 7: Cumulative Distribution Function (CDF) results comparison of Cloud Computing (Cloud), Edge Computing (at drone and GCS), Edge with High Performance Computing (Edge HPC) and Our Approach architectures in terms of: (a) accuracy rate on car counting, (b) Packet processing Frame-per-Second (FPS), and (c) Tracking rate.



(a) (b) (c)

Figure 8: Computation off-loading choices preferred amongst Cloud Computing (Cloud), Edge Computing (at drone and GCS), Edge with High Performance Computing (Edge HPC) and Our Approach architectures in terms of: (a) accuracy rate on car counting, (b) Tracking rate, and (c) packet processing Frame-per-Second (fps).

conditions, irrespective of the type of occlusion and the networking protocol used for the video transmission.

The occlusion rate results corresponding to stalling and blurring for various available network bandwidth conditions is shown in Figure 6c. In this set of experiments, we use live streaming the between source (drone) and the destination (GCS) for various available network bandwidth conditions. A similar setup is repeated for networking protocols such as TCP, UDP, RTP and QUIC. We can conclude from the results shown in Figure 6c that - under high-speed network bandwidth connections, the occlusion rate decrease irrespective of its type. In specific conditions with bottleneck network link bandwidth conditions with occlusions, we find that the suggested protocol for satisfactory video streaming performance is obtained for the QUIC protocol with 420 pixels in the object recognition application context, and for the UDP protocol with 420 pixels in the object tracking application context.

5.2.3 Dynamic Decision Making Scheme Results. In the first set of results from Figure 7b, we find that our dynamic decision making scheme is an optimal choice w.r.t. tracking rate and accuracy on object detection factors, if real-time guidance is needed. From the Figure 7b, we can see that the drone level processing can process only at a processing speed of ≤ 0.01 FPS, which is not suitable for computation purpose in real time application contexts. However, drone level processing is suited for decision making purposes. In comparison, the proposed FCC approach has a processing speed of $\geqslant 0.03$ FPS, when compared with the edge GCS and cloud architectures. Thus, we can conclude that our proposed FCC scheme is an optimal choice w.r.t tracking rate and accuracy.

In our second set of results from Figures 7a and 7c, we find that our dynamic decision making scheme is superior than the drone-only, edge-GCS and cloud-only cases in terms of the processing speed. By sacrificing 20% of accuracy and tracking rate for the frame processing, our proposed scheme performs the best for the video analytics application considered. From Figures 7a and 7c, we can conclude that the choice of drone-only or edge-GCS is a suitable when there are cases of network failure that causes occlusion detection. Our proposed scheme achieves 25% accuracy improvement and 82% improvement on tracking rate than the other approaches by limiting the resources consumed by switching to a low quality of video frame resolution.

In our third and final set of results from Figures 8a, 8b and 8c, we find that our dynamic decision making scheme has the best performance based on occlusion detection under bottleneck network conditions, and in terms of the processing speed. In the first stage, the network connection quality is estimated and accordingly the decision to off-load the computation is done dynamically. If the response time is less, a portion of the function/task is offloaded onto FCC or it is stored in the local drone buffer. The proposed dynamic off-loading scheme shows improvement over the other offloading strategies under a variety of bottleneck network conditions. First, if a network failure occurs at the beginning of the application pipeline or during the video streaming, our proposed scheme will check if the *FCC* policy can be used. If so, a set of pre-processing functions in drone are invoked that utilize the drone's on-board computation resources. Alternately, the video stream is buffered on the drone storage to avoid packet/information loss, which in turn improves the tracking rate as well as the accuracy rate. On the contrary, when

we are concerned about the accuracy and tracking rate of the object, both *drone-only* and our proposed scheme are suitable choices.

6 CONCLUSION

In this paper, we presented a novel dynamic computation off-loading and control scheme that detects occlusions and coordinates intelligent processing in drone video analytics. Our scheme takes into account not only the computation workload and processing time, but also the dynamic parameters that affect drone video analytics performance in a real-time manner. We handle network bottlenecks with regards to drone mobility, data transmission protocol in video streaming, and various occlusion types resulting from changes in these dynamic parameters. Our evaluation results collected from a near realistic testbed show how our dynamic computation off-loading scheme can intelligently balance the trade-offs between tracking/accuracy rate vs. processing time using a novel Function-centric Computing (FCC) architecture with regards to different networking protocols under different available bandwidth network connections between the drones and a GCS. Our results also demonstrated how the FCC architecture efficiently utilizes the drone edge, GCS edge and cloud computing resources while optimally offloading any decomposable application functions/tasks in the form of microservices among these resources. Lastly, we demonstrated object recognition and tracking results with our approach under different types of occlusion rates measured using PSNR thresholds. Specifically, we showed how our approach can significantly improve users' QoE levels in terms of tracking/accuracy within drone-based video streaming services as part of the video collection/analysis steps of an object detection pipeline application.

Future work could focus on performance evaluation studies on handling additional QoE factors related to the occlusion detection on drone captured videos, e.g., blocking of the object of interest by other objects and night vision based occlusions due to lack of visible light. In the same context, additional system parameters can be handled that could impact computation off-loading decision making in a real scenario, e.g., energy consumption on the drone devices in object tracking and/or geo-location reconstruction.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation (NSF) under Award Number: CNS-1647182. It is also supported by the Army Research Lab (ARL) under Award Numbers: W911NF1820285 and W911NF1910181. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the NSF or the ARL.

REFERENCES

[1] I. Bekmezci, I. Sen, and E. Erkalkan. 2015. Flying ad hoc networks (FANET) test bed implementation. In *2015 7th International Conference on Recent Advances in Space Technologies (RAST)*. IEEE, 665–668.

[2] P. Calyam, P. Chandrasekaran, G. Trueb, N. Howes, R. Ramnath, D. Yu, Y. Liu, L. Xiong, and D. Yang. 2012. Multi-Resolution Multimedia QoE Models for IPTV Applications. *Int. J. Digital Multimedia Broadcasting* 2012 (2012), 904072:1–904072:13.

[3] P. Calyam, M. Haffner, E. Ekici, and C.G.Lee. 2007. Measuring Interaction QoE in Internet Videoconferencing. In *Real-Time Mobile Multimedia Services*, Dilip Krishnaswamy, Tom Pfeifer, and Danny Raz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 14–25.

[4] D. Chemodanov, P. Calyam, and F. Esposito. 2019. A Near Optimal Reliable Composition Approach for Geo-Distributed Latency-Sensitive Service Chains. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 1792–1800. https://doi.org/10.1109/INFOCOM.2019.8737498

[5] D. Chemodanov, P. Calyam, S. Valluripally, H. Trinh, J. Patman, and K. Palaniappan. 2018. On QoE-oriented Cloud Service Orchestration for Application Providers. *IEEE Transactions on Services Computing* (2018), 1–1. https://doi.org/10.1109/TSC.2018.2866851

[6] D. Chemodanov, C. Qu, O. Opeoluwa, S. Wang, and P. Calyam. 2019. Policy-Based Function-Centric Computation Offloading for Real-Time Drone Video Analytics. In *2019 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. 1–6. https://doi.org/10.1109/LANMAN.2019.8847112

[7] S. Chinchali et al. 2019. Network Offloading Policies for Cloud Robotics: a Learning-based Approach. *CoRR* abs/1902.05703 (2019). arXiv:1902.05703 http://arxiv.org/abs/1902.05703

[8] P. Duygulu and H. Wactlar. 2003. Associating video frames with text.

[9] K. Gatimu, A. Dhamodaran, T. Johnson, and B. Lee. 2018. Experimental study of low-latency HD VoD streaming using flexible dual TCP-UDP streaming protocol. In *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*. 1–6. https://doi.org/10.1109/CCNC.2018.8319234

[10] C. Grasso and G. Schembra. 2019. A Fleet of MEC UAVs to Extend a 5G Network Slice for Video Monitoring with Low-Latency Constraints. *Journal of Sensor and Actuator Networks* 8 (2019).

[11] A. Guillen-Perez and M. Cano. 2018. Flying ad hoc networks: A new domain for network communications. *Sensors* 18, 10 (2018), 3571.

[12] I.Bekmezci, I. Sen, and E. Erkalkan. 2015. Flying ad hoc networks (FANET) test bed implementation. In *2015 7th International Conference on Recent Advances in Space Technologies (RAST)*. IEEE, 665–668.

[13] W. Jung, J. Yim, Y. Ko, and S. Singh. 2017. ACODS: adaptive computation offloading for drone surveillance system. In *2017 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. 1–6. https://doi.org/10.1109/MedHocNet.2017.8001647

[14] B. Kim et al. 2018. Dynamic Computation Offloading Scheme for Drone-Based Surveillance Systems. *Sensors* 18 (09 2018), 2982. https://doi.org/10.3390/s18092982

[15] A. Mahdi et al. 2017. Coverage Maximization for a Poisson Field of Drone Cells. https://doi.org/10.1109/PIMRC.2017.8292753

[16] B. Mark, J Chase, and Lawrence Landweber. 2014. GENI: A federated testbed for innovative network experiments. *Computer Networks* (2014).

[17] M. Mark. 2011. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces.* " O'Reilly Media, Inc.".

[18] M. H. Naser, T. Tarik, and A. Osama. 2016. Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives. *IEEE Internet of Things Journal* 3, 6 (2016), 899–922.

[19] Nvidia. 2019. NVidia jetson - The AI Platform for Autonomous Machines. https://developer.nvidia.com/embedded/develop/hardware

[20] O. Opeoluwa. 2019. Moving Object Classification Application Repository. https://github.com/Blowoffvalve/ImageProcessingWebServices.

[21] J. Redmon, S. Divvalaand, et al. 2016. You only look once: Unified, real-time object detection. In *IEEE CVPR*. 779–788.

[22] H. Schulzrinne, C. Stephen, R. Frederick, and V. Jacobson. 2003. RTP: A Transport Protocol for Real-Time Applications. *Internet Engineering Task Force, RFC* (07 2003).

[23] M. Seufert, R. Schatz, N. Wehner, and P. Casas. 2019. QUICker or not? -an Empirical Analysis of QUIC vs TCP for Video Streaming QoE Provisioning. In *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. 7–12. https://doi.org/10.1109/ICIN.2019.8685913

[24] T. Soyata et al. 2012. Cloud-Vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In *2012 IEEE Symposium on Computers and Communications (ISCC)*. 59–66. https://doi.org/10.1109/ISCC.2012.6249269

[25] A. Sukhov, P. Calyam, W. Daly, and A. Illin. 2005. Towards an Analytical Model for Characterizing Behavior of High-Speed VVoIP Applications. In *TNC*.

[26] H. Trinh, P. Calyam, D. Chemodanov, S. Yao, Q. Lei, F. Gao, and K. Palaniappan. 2018. Energy-Aware Mobile Edge Computing and Routing for Low-Latency Visual Data Processing. *IEEE Transactions on Multimedia* 20 (08 2018), 1–1. https://doi.org/10.1109/TMM.2018.2865661

[27] R. Valentinoand W. Jungand Y. Ko. 2018. A Design and Simulation of the Opportunistic Computation Offloading with Learning-Based Prediction for Unmanned Aerial Vehicle (UAV) Clustering Networks. *Sensors* 18 (11 2018), 3751. https://doi.org/10.3390/s18113751

[28] L. Xie et al. 2004. Discovering meaningful multimedia patterns with audio-visual concepts and associated text. In *2004 International Conference on Image Processing, 2004. ICIP '04.*, Vol. 4. 2383–2386 Vol. 4. https://doi.org/10.1109/ICIP.2004.1421580

[29] T. Yu-Han Chen et al. 2016. GLIMPSE: Continuous, Real-Time Object Recognition on Mobile Devices. *GetMobile: Mobile Comp. and Comm.* 20, 1 (July 2016), 26–29.

[30] P. Zhu et al. 2018. Vision Meets Drones: A Challenge. *arXiv preprint arXiv:1804.07437* (2018).