Performability Modeling and Analysis for Real-Time Object Detection on UAV Systems

Qingyang Zhang
Department of Computer Science
University of Tsukuba
Tsukuba, Japan
zhang.qingyang@sd.cs.tsukuba.ac.jp

Fumio Machida
Department of Computer Science
University of Tsukuba
Tsukuba, Japan
machida@cs.tsukuba.ac.jp

Ermeson Andrade
Department of Computing
Federal Rural University of Pernambuco
Recife, Brazil
ermeson.andrade@ufrpe.br

***Abstract*—With the widespread application of Uncrewed Aerial Vehicles (UAVs) in various real-world surveillance scenarios, the quality analysis of UAV-based monitoring systems has become an emergent challenge. Previous model-based studies often made theoretical assumptions to estimate the performance and availability of UAV computing systems, without detailed considerations of the interaction between UAVs and fog computing nodes, as well as computation steps of object detection algorithms. In this paper, we propose Stochastic Reward Nets (SRNs) to capture computational behavior and analyze performance, availability, and performability metrics of a UAV system that utilizes computation offloading. In order to obtain more realistic parameters for model-based analysis, we conduct empirical experiments using an edge computing device to emulate real-time object detection on a UAV. We measure the throughput of real-time object detection process in three stages by experiments that are fed into parameters for numerical analysis on the proposed model. Through the sensitivity analysis, we demonstrate the impact of different computation modes and video resolutions on performance and availability metrics, providing insights for improving UAV system design and operation.**

***Keywords*—Object detection, Performability, Stochastic reward nets, Uncrewed aerial vehicle.**

I. Introduction

In recent years, Uncrewed Aerial Vehicles (UAVs), also known as drones, have garnered widespread attention from both practitioners and researchers due to their advanced functionalities, scalability in hardware and software, user-friendly operation, and cost-effectiveness [1], [2]. However, UAV systems are susceptible to various factors impacting their performance. It is essential to understand how system performance, such as service availability and task throughput, are influenced by network accessibility, interactions with other computing nodes, and the resolutions of video frames.

Performability analysis and design of UAV-based systems are becoming a challenging issue. Performability is the composite measure of performance and availability, which can be quantitatively evaluated through the performance and availability models and measurements [3]. These quality metrics are often intercorrelated and subject to trade-off relations under operation environments [4]. Given that UAVs, as mobile edge devices, have limited computing resources and battery life, enhancing performability becomes a fundamental requirement because it directly impacts mission success, operational efficiency, and overall system effectiveness.

One major limitation of UAVs is their restricted resources. To deal with this, computation offloading is commonly used. This technique involves migrating tasks from the UAV's onboard resources to remote servers [5]. While performance models play a crucial role in evaluating the effects of offloading [6]–[8], prior research often lacks real-world data, relying on hypothetical values, further limiting their applicability and accuracy.

In this paper, we use Stochastic Reward Nets (SRNs) [9] to develop a model for UAV-based monitoring systems, which comprehensively incorporates the behaviors of object detection algorithms and task offloading between the drone and the fog node. Unlike previous studies, we conduct experiments to profile object detection algorithm performance, enhancing parameter estimation accuracy. Evaluating YOLO (You Only Look Once) v5 [10] and YOLOv3 [11] on edge and fog processing modes using emulated devices, we collect performance metrics for SRN parameterization. Our numerical sensitivity analysis indicates that the choice of different computation modes and video resolutions significantly influences the service availability, service throughput, and performability of the UAV system, all of which are affected by the failure rate of the computing process.

We make the following contributions in this paper:

- Introduction of comprehensive SRN models for analyzing performance and availability of real-time object detection system on UAVs with a fog node serving as an offloading server.
- Experimental evaluation of two real-time object detection algorithms on an edge computing device to obtain realistic parameter values for numerical analysis on the proposed models.
- Insightful sensitivity analysis results that show the impact of computation mode and video resolution choices on performance metrics such as service availability, system throughput, the frame drop ratio, and performability.

The rest of the paper is organized as follows. Section II describes related work. Section III introduces the target system. Section IV presents the SRNs to evaluate the performance and availability considering two computation modes. Section

V shows our experimental results on an emulated computing device. Section VI illustrates results of sensitivity analysis about two computation mode. Finally, in Section VII, we provide our conclusion.

II. Related Work

Recent studies exploit object detection in UAV surveillance systems using diverse models and hardware implementations. Ajith et al. [12] introduced a unique hybrid deep learning model for object recognition that can aid in search and rescue operations. Chen et al. [13] presented a system that combines an FPGA and a drone with a neural-network engine for real-time object detection, achieving high FPS and lower power consumption. While these studies improved the performance of object detection in UAV, the interactions between a UAV, a control terminal, and a node to offload the object detection tasks were not considered. Our experiments and model-based study follow these interactions to evaluate the system performance.

Many researchers have evaluated the performance of UAV application systems. Wang et al. [14] assessed the performance and operation failure rate of agricultural drones. The experimental statistical results showed that the drone made failure about 4% of the time, and the net working time was only 30%, revealing the inefficiency of monitoring caused by unavailability. Petritoli et al. [15] addressed reliability issues in UAV design and identified constraints to ensure appropriate preventive maintenance intervals. These studies used statistical analysis or probability functions to evaluate the performance of UAVs. In our study, we introduce a comprehensive SRN-based performability model for a fog-assisted drone monitoring system to model and analyze the performance under different computational modes.

Sensitivity analysis is one of the essential methods to analyze the system performance bottleneck. Existing studies have utilized Stochastic Petri nets (SPNs) to model systems and analyze the sensitivity of system parameters to their performance. Zhang et al. [7] identified a bottleneck in system performance caused by multiple UAVs competing for fog node computing resources, then proposed improvement suggestions based on sensitivity analysis results. Sabino et al. [16] proposed a fire monitoring system using drones and edge computing, assessing its structure with SPNs to evaluate the impacts of queue capacity, processor cores, and service duration. In contrast to these works, our study examines the detailed performance of each step in object detection under two computation modes with different video resolutions and quantifies system performability.

III. UAV-based monitoring system

In this section, we outline our target system configuration and provide a general overview of the steps involved in a single-stage object detector.

A. System configuration

In our study, we examine a scenario involving a UAV equipped with a camera and a computational unit for video processing. During its flight, the UAV has the option to perform local object detection processing onboard or offload the captured video footage to a computing node for processing. We assume that computing nodes within a fog computing infrastructure is accessible through wireless networks [8], as shown in Figure 1. Upon allocation of an available fog node, the UAV establishes a connection and transmits the image data for processing, with the resulting object detection outcome returned to the UAV. It is important to note that successful offloading relies on a stable wireless communication link. In cases where environmental conditions disrupt communication stability, offloading becomes impractical.

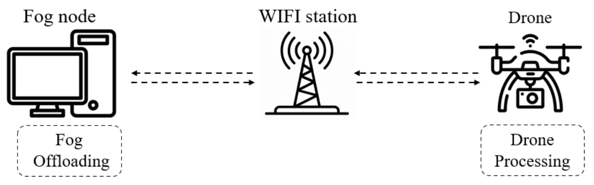


Fig. 1: Drone computation system

B. General steps of a single-stage object detector

In recent years, advancements in single-stage detectors have led to their performance in target detection tasks approaching or even surpassing that of two-stage detectors [17] in real-time object detection, such as the YOLO series [18] and EfficientDet [19]. The general steps of a single-stage object detector can be divided into three stages, preprocessing, inference and NMS (Non-Maximum Suppression):

Preprocessing: The input image undergoes resize and normalization, with optional data augmentation techniques like random cropping and flipping.

Inference: Convolutional Neural Networks (CNNs) extract features from the preprocessed image, enabling object detection and classification, which generate candidate bounding boxes. Some detectors, such as CenterNet [20], may bypass the NMS stage if candidate boxes are directly output.

NMS Stage: NMS is applied to filter and merge overlapping candidate bounding boxes, thereby obtaining the final detection results.

While specific implementation details may vary among single-stage detectors, they all follow this fundamental process. In our study, we employ YOLOv3-tiny [11] and YOLOv5s [10] as experimental examples.

IV. Propsed Model

This section details the SRNs for evaluating the performability of an image processing and task offloading system considering one UAV and one fog node. First, we briefly introduce the formalism of SRNs.

A. Stochastic Reward Nets (SRNs)

Petri nets are formalisms for modeling different types of systems, from cyber-physical systems to communication protocols. A Petri net is a directed bipartite graph comprised of two types of nodes: places, represented by circles, and transitions, represented by rectangles (see Figure 2). Places are connected to transitions via incoming or outgoing arcs. A marking of a Petri net is represented by the number of tokens in its places, with each marking corresponding to a specific system state. State transitions are indicated by changes in marking resulting from the firing of a transition. A transition becomes enabled when all input places have the required number of tokens. Upon firing, a transition removes tokens from its input places based on specified multiplicities and deposits new tokens into its output places.

Inhibitor arcs are special arcs used to restrict transition enablement conditions. As depicted in Figure 2, an inhibitor arc is illustrated as a line ending in a small circle. A transition remains disabled when a limited number of tokens is present in the connected place via an inhibitor arc, here is one token.

The original Petri net model lacks the concept of time, which is essential for analyzing performance and availability. The introduction of time results in a type of Petri net called timed Petri nets, with one specific variant used in this study being SRNs [22]. Within SRNs, two types of transitions are considered: timed and immediate. Timed transitions are associated with exponentially distributed firing times, while immediate transitions fire instantaneously. Additionally, SRNs incorporate guard functions to manage transition conditions and reward functions to facilitate the computation of performance measures. Software packages such as SPNP [21] and SHARPE [22] support the solution of SRNs.

In this paper, we assume that all timed transitions have exponentially distributed firing times. Our model is extended from the SRN originally presented in [6]. Further details about SRN formalism, solution techniques and modeling examples can be found in [23].

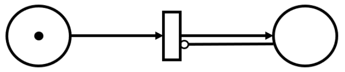


Fig. 2: An example of Petri Net with an inhibitor arc

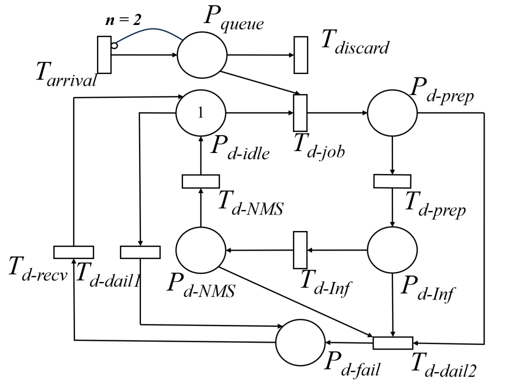
B. Performability computation

Performability, which combines performance and availability, provides a comprehensive evaluation of a system's overall operational effectiveness. Many practical application systems require not only high-performance but also need high-availability during their operation. A high performance but unavailable system may not meet users' expectations, while a highly available system with low performance is not acceptable either [8].

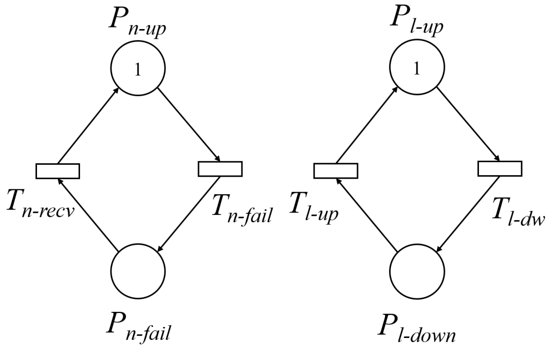
In the context of a drone image processing system, our focus lies on evaluating the request processing capability during its operational duration. The average request processing rate throughout the available period serves as a key performability metric of interest. Thus, we follow the definition of performability presented in [6]. Let s be a state of a drone system and denote $p(s)$ as the performance measure (which, in this work, refers to task throughput) when the system is in state s. The performability measure can be defined as

$$P_a = \int_{s \in A} p(s) dF(s) \tag{1}$$

where A represents the set of states where the system is available and $F(s)$ is the probability distribution of the random variable for the state s. To compute the value of P_a, the expected performance for each state s, as well as the probability of the state s needs to be estimated. These values are not easily given from the system specification since state changes are induced by environmental uncertainties. Therefore, we introduce stochastic models to capture the state transitions of the system under environmental uncertainties.



(a) Edge model



(b) Client node model (c) Wirless link model

Fig. 3: SRN for the edge processing mode

C. Edge processing (EP) model

Fig 3 presents the SRN for the drone image processing system, which includes an edge device, client node model, and wireless model

The edge model shown in Fig 3a represents the states of the job processing on the edge device. T_{arrival} represent the obtained video stream with rate γ_{arr}. Incoming video frames are saved in P_{queue}, but the inhibitor arc ensures that only two tokens are in memory waiting for processor allocation for the subsequent object detection stage. Video frames that cannot be processed in time due to insufficient system performance will be discarded, that is, T_{discard} fires. Therefore, what is retained in P_{queue} are frames processed for real-time video streaming with rate γ_{disc}.

Initially, the edge is idle, represented by a token in $P_{\text{d-idle}}$. Image processing requests, modeled by a Poisson process with rate γ_{in}, transition the system to the preprocessing state, a token is deposited in $P_{\text{d-pre}}$, upon firing the $T_{\text{d-job}}$. Subsequent transitions $T_{\text{d-prep}}$, $T_{\text{d-inf}}$, and $T_{\text{d-NMS}}$ represent preprocessing, inference, and non-maximum suppression stages, respectively. Transition rates $\nu_{\text{d-prep}}$, $\nu_{\text{d-inf}}$, and $\nu_{\text{d-nms}}$ represent the processing speeds for each stage: the preprocessing, inference, and NMS, respectively.

During the operation, the computation process may encounter a failure. Failures during processing are accounted for by transitions $T_{\text{d-fail1}}$ and $T_{\text{d-fail2}}$, triggered in idle and processing state failures, respectively. The process recovery transition $T_{\text{d-recv}}$ fires when recovering a failure with rate μ_d.

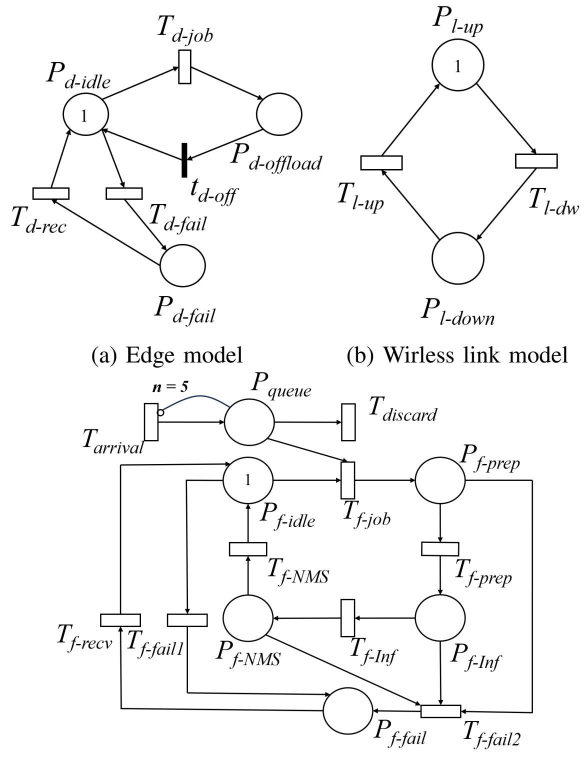
The client node model in Figure 3b captures the failure and recovery behavior of the client that receives processing results from the drone. Although the client state does not affect the drone's process, users cannot access results when the client is unavailable. Hence, we consider the client's state to calculate service availability. A token in $P_{\text{n-up}}$ is removed when $T_{\text{n-fail}}$ fires, that is a node failure event. Meanwhile, a token in $P_{\text{n-fail}}$ is removed when $T_{\text{n-rec}}$ fires, representing node recovery event. We assign failure rate λ_n and recovery rate μ_n to $T_{\text{n-fail}}$ and $T_{\text{n-rec}}$, respectively.

The wireless link model in Figure 3c represents the communication link state between the drone and the client. Even if both operate properly, link disconnection causes system unavailability. A token in $P_{\text{l-up}}$ enables $T_{\text{l-down}}$, representing link disconnection. While a token in $P_{\text{l-down}}$ enables $T_{\text{l-up}}$, representing link reconnection. Link failure and reconnection rates are λ_l and μ_l, respectively.

In this paper, we consider the three performance metrics for EP mode.

Service availability is defined as the probability of the system being in non-failure states. In the edge processing mode, the service is available unless a token is deposited in $P_{\text{d-fail}}$, $P_{\text{n-fail}}$ and $P_{\text{l-down}}$.

Service throughput is also considered a performance measure of an image processing system and can be computed by the product of the probability of the processing state and the corresponding service rate [6]. As the throughput of object detection process in a failure-free execution can be measured by FPS (Frames Per Second) through the experiments, we use the collected values to estimate the effective service rates of the edge node and the fog node. By breaking down the processing time into individual stages and measuring their FPSs, we gain insights into the performance bottlenecks of each stage. In our SRN, the expected service rate is the rate of $T_{\text{d-job}}$.



(a) Edge model (b) Wirless link model

(c) Fog node model

Fig. 4: SRN for the fog processing mode

Frame drop ratio is defined as the proportion of discarded frames to the total number of frames requested. Under different factors impacts, this value will represent the completion rate of the service.

To compute the above performance metrics, we define the reward functions as shown in Table I.

D. Fog processing (FP) model

For the fog computing mode, we consider one UAV and one fog node that are used when computation offloading is requested. Figure 4 shows the SRN that consists of interrelated subnets representing the edge node, fog node, and wireless link for communication. Table II shows the guard functions assigned to the transitions of the subnets.

In the fog offloading mode, the edge model includes the place $P_{\text{d-offload}}$ to represent the state to request offloading. When a token is deposited in $P_{\text{d-idle}}$ after firing and the guard condition *gd_off* is met, the transition $T_{\text{d-job}}$ is enabled and a new token is deposited in $P_{\text{d-offload}}$ with rate γ_{arr}. Subsequently, when the conditions of the guard function *gf_job* are met, and T_{arrival} is fired, indicating that a task has been sent to the fog node.

When the token in P_{queue} is non-zero, and the condition of the guard function *gd_comp* is met, the immediate transition $t_{\text{d-off}}$ is enabled, which means that some tasks have been offloaded to the fog node. After immediate transition $t_{\text{d-off}}$ fires, the token in $P_{\text{d-offload}}$ is removed, and a new token is generated in $P_{\text{d-idle}}$, indicating that data has been sent to the

TABLE I: Reward function for SRNs

Measure	Function
Edge processing	
Service availability	($\#P_{\text{d-idle}}$ == 1 or $\#P_{\text{d-prep}}$ == 1 or $\#P_{\text{d-Inf}}$ == 1 or $\#P_{\text{d-NMS}}$ == 1) and ($\#P_{\text{n-up}}$ == 1) and ($\#P_{\text{l-up}}$ == 1)) then 1 else 0
Service throughput	prob($\#P_{\text{d-idle}}$ == 1) * γ_{in} (i.e. rate("$T_{\text{d-job}}$"))
Frame drop rate	rate("T_{discard}")/rate("T_{arrival}")
Fog processsing	
Service availability	($\#P_{\text{d-idle}}$ == 1 or $\#P_{\text{d-offload}}$ == 1) and ($\#P_{\text{f-idle}}$ == 1 or $\#P_{\text{f-prep}}$ == 1 or $\#P_{\text{f-Inf}}$ == 1 or $\#P_{\text{f-NMS}}$ == 1)) and ($\#P_{\text{l-up}}$ == 1)) then 1 else 0
Service throughput	prob($\#P_{\text{f-idle}}$ == 1) * γ_{in} (i.e. rate("$T_{\text{f-job}}$"))
Frame drop ratio	rate("T_{discard}")/rate("T_{arrival}")

fog node and the drone returns to the idle state. We assume that the average time to start an task transmission in the fog node due to communication delay is δ in T_{arrival}. During the fog node processing, the process may finish the job or fail, which are represented by the transitions $T_{\text{f-prep}}$, $T_{\text{f-NMS}}$ or $T_{\text{f-dail}}$, respectively.

For the fog processing model, we consider service availability, throughput and frame drop ratio as the performance metrics. The reward functions for these metrics are defined as shown in Table II. For the Service availability, the service is available unless a token is deposited in $P_{\text{d-fail}}$, $P_{\text{f-fail}}$ and $P_{\text{l-down}}$. Service throughput is measured by the expected service rate of $T_{\text{f-job}}$.

TABLE II: Guard function for the fog offloading mode

Name	Transition	Function
gd_off	$T_{\text{d-job}}$	if ($\#P_{\text{l-up}}$==1) then 1 else 0
gf_job	T_{arrival}	if ($\#P_{offload}$==1) then 1 else 0
gd_comp	$t_{\text{d-off}}$	if ($\#P_{queue}\geq$0) then 1 else 0

V. Experiment

The SRN models presented in the previous section need parameter values for numerical analysis. To estimate a realistic parameter values, we conduct experiments and collect performance data of object detection algorithms.

A. Experiment design

1) Flight simulation & record: We use AirSim to create a simulation environment with featuring mountains and UAV models. In the simulated environment covering 30,000 square meters, we deployed 16 deer and 14 wolf models. The objective of the UAV flights is to conduct surveillance of animal activities and monitor the ecological environment within the specified range. Video streams are captured by the camera on the UAV in AirSim.

While the video we recorded from AirSim had a resolution of FHD (1980x1080) and ran at 30 FPS, the video stream is rebroadcasted in the RTMP (Real Time Messaging Protocol) server at various resolutions, including FHD, and HVGA (480×270) resolution, with a frequency of 30 FPS for each experiment. One of our flights lasts approximately six minutes. Figure 5 shows some sample images of the simulated environment captured by the camera.

Fig. 5: Samples of forest images captured by AirSim

2) Experimental setup: To replicate the animal detection process on the UAV, we set up an experimental system consisting of a computing device serving as a video capture terminal and an RTMP server for streaming recorded videos. Connected to a local area network via Wi-Fi 802.11ax, a PC acts as a fog node to evaluate the performance of the offloading mode, denoted as fog processing.

For image processing on the UAV, we emulate it using a Raspberry Pi 4B as an edge device, powered by a battery supply to simulate real-world conditions. This computational mode is referred to as edge processing. The CPU processing mode is employed for image processing on the Raspberry Pi, considering its shared memory architecture and higher CPU frequency compared to the GPU. Similarly, to ensure a fair comparison of performance, we employ CPU-based detection processing on the fog node as well.

Figure 6 illustrates the system configuration utilized in our experiment. The hardware and software configurations are detailed as follows.

- Edge Node: Raspberry Pi 4B with Debian GNU/LINUX 11 OS;
- Edge node external power supply: PiSugar 3 Plus, 5000mAh;
- Fog Node: ASUS ROG Strix G512LV314 with Windows 10 Home OS;
- RTMP server/Video collector: ASUS VivoMini VC65 with Ubuntu 18.04.6 TLS.

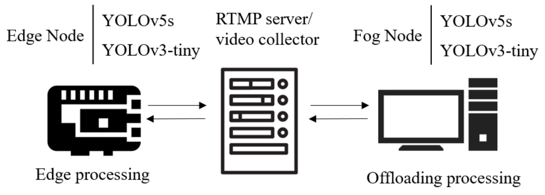


Fig. 6: Experimental configuration for performance measurements

Next, the performance of a real-time object detection system are measured on the edge device and the fog node computer. During the execution of the target detection algorithm on the input video stream, we recorded the processing time for each frame across three stages of the image processing pipeline: preprocessing, inference, and NMS.

B. Edge processing

We conduct three experiments for each video input resolution (FHD and HVGA) and take the average of the observed values. The results are summarized in Tables III and IV.

As anticipated, the FPS of overall processing increases with lower resolution images, with HVGA resolution cases yielding the highest FPS. During object detection processing, each frame is processed in a fraction of a second. If the processing of a frame extends beyond the arrival of the subsequent frame, it necessitates additional time to process the subsequent frame. Consequently, FPS diminishes with larger image sizes, requiring prolonged processing durations. Moreover, the processing speed is observed to be higher when employing YOLOv3 compared to YOLOv5. The smaller model size of YOLOv3-tiny contributes to this enhancement, although without explicit consideration of potential accuracy rate reductions. In subsequent performability analyses, we solely examine the impact of changes in system performance on processing speed.

TABLE III: Object detection performance in EP (YOLOv5)

Resolution	Processing speed	Preprocessing	Inference	NMS	Total
FHD	Time/ms	13.7467	3853.24	3.3339	3870.321
	FPS	72.7447	0.2595	299.9453	0.2584
HVGA	Time/ms	8.4625	1124.667	1.3083	1134.438
	FPS	118.1687	0.8892	764.3471	0.8815

TABLE IV: Object detection performance in EP (YOLOv3)

Resolution	Processing speed	Preprocessing	Inference	NMS	Total
FHD	Time/ms	14.4057	1952.3191	7.5463	1974.2711
	FPS	69.4170	0.5122	132.5158	0.5065
HVGA	Time/ms	7.9920	723.2023	2.0202	733.2144
	FPS	125.1258	1.3827	495.0046	1.3639

C. Fog processing

Subsequently, we execute real-time video detection by re-broadcasting the recorded video to the fog node through the RTMP server, with two different resolutions of video streams.

The performance outcomes of the object detection process at the fog node under stable network conditions are detailed in Table V and VI. Notably, the FPS values exhibit significant increase compared to those observed in EP mode, attributable to the higher computational capabilities of the fog node. Similarly, when we use YOLOv3, the processing speed surpasses that of YOLOv5. In contrast to the edge processing mode, no discernible correlations between image resolutions and FPS values are observed in the fog processing mode.

TABLE V: Object detection performance in FP (YOLOv5)

Resolution	Processing speed	Preprocessing	Inference	NMS	Total
FHD	Time/ms	0.6881	102.1139	0.3476	103.1496
	FPS	1453.1813	9.7930	2877.1140	9.6947
HVGA	Time/ms	0.8237	101.0870	0.8159	102.7265
	FPS	2162.4353	9.8697	3698.6470	9.7989

TABLE VI: Object detection performance in FP (YOLOv3)

Resolution	Processing speed	Preprocessing	Inference	NMS	Total
FHD	Time/ms	0.6368	69.0959	0.5639	70.2966
	FPS	1570.3214	14.4726	1773.4900	14.2254
HVGA	Time/ms	0.9919	64.0807	0.5292	65.6019
	FPS	2393.0554	16.0307	1861.1170	15.7890

D. Comparison

We segmented an object detection algorithm into three stages and measured the average speeds. Regardless of the computing device, the preprocessing speeds of HVGA input by YOLOv5s and YOLOv3-tiny were approximately 1.5 to 1.8 times faster than those of FHD input. This ratio tends to remain consistent, as the time complexity of the image preprocessing stage can be considered linear (O(n)). Overall, the preprocessing speed of the YOLOv3-tiny model is slightly faster than that of YOLOv5s, possibly due to optimizations for hardware.

During the inference stage, YOLOv3-tiny exhibits higher speeds than YOLOv5s, as the model parameters are smaller, making it more suitable for edge device utilization. However, due to the limited resources in the edge node, there is no significant difference in inference speeds between the two algorithms for both resolutions. Meanwhile, the difference in inference speeds in the fog node is significant. Similar observations are also found in the NMS stage.

VI. Numerical analysis

In this section, we conduct numerical experiments with the estimated performance parameter values from the experiments. We compare the service availability, service throughput, frame drop ratio, and performability achieved by two computation modes with two different video resolutions.

A. Parameterization

We follow the baseline parameters used in the previous study [6]. Table VII shows the parameters and their default values used in the experiments. The process can fail more frequently in the processing states. Therefore the model has also transitions $T_{\text{d-fail2}}$ and $T_{\text{n-fail2}}$ that represent the process failure events during the job execution in the edge processing mode and fog offloading mode, respectively. We vary the λ_d in range of [0, 0.1] (1/hour). The failure rate in the processing states are set to $1.5\lambda_d$, as it should be larger than that in the idle state. For the failure rate of the fog node during idle and processing states, fixed default values are employed. For the service rate, we use the collected data of FPSs from the experiments.

TABLE VII: Parameters for Numerical Experiment

Variable	Description	Value
γ_{arr}	Task arrival rate in edge processing	30
γ_{in}	Processing request rate	2400
$\nu_{d\text{-}prep}$	Preprocessing rate on an edge device	$\nu_{d\text{-}inf}$ in Table III and IV
$\nu_{d\text{-}inf}$	Inference rate on an edge device	$\nu_{d\text{-}prep}$ in Table III and IV
$\nu_{d\text{-}NMS}$	NMS rate on an edge device	$\nu_{d\text{-}NMS}$ in Table III and IV
$\nu_{n\text{-}prep}$	Preprocessing rate on a fog node	$\nu_{n\text{-}inf}$ in Table V and VI
$\nu_{n\text{-}inf}$	Inference rate on a fog node	$\nu_{n\text{-}prep}$ in Table V and VI
$\nu_{n\text{-}NMS}$	NMS rate on a fog node	$\nu_{n\text{-}NMS}$ in Table V and VI
μ_d	Process recovery rate on a drone	3 [6]
μ_n	Process recovery rate on a fog node	2 [6]
δ	Task transmission rate	7200 [6]
λ_n	Failure rate on a fog node in the idle state	0.000462963 [6]
λ_{n2}	Failure rate on a fog node in the processing state	0.001388889 [6]
λ_l	Communication link disconnection rate	0.5 [6]
μ_l	Communication link reconnection rate	360 [6]

B. Sensitivity analysis

Figures 7 and 8 show the results of sensitivity analysis of λ_d on the service availability, task throughput, frame drop ratio, and performability for YOLOv5 and YOLOv3, respectively. In each plot, we compare the results of two computation modes with different video resolutions.

As shown in Figures 7a and 8a, regardless of the choice of detector, when the failure rate at the edge node is lower than that of the fog node, the service availability of EP tends to be slightly higher than FP. However, when the the failure rate increases, the advantages of FP mode become more obvious, and it achieves higher availability compared to EP. In terms of resolution, there is not much difference in availability between the same computing modes.

Regarding throughput, as illustrated in Figures 7b and 8b, regardless of the algorithm employed, the variation trends are primarily evident across different computation modes rather than the choice of resolution. Specifically, in the case of FP mode with YOLOv5, the FPS remains relatively consistent across both resolutions, exhibiting a linear decline as the failure rate increases. On the other hand, in the EP mode, the decline is less pronounced. With YOLOv3, distinct FPS values are observed across both resolutions within the FP mode, yet the downward trend remains consistent as the failure rate increases. Similarly, the EP mode displays a similar trend as in the case of YOLOv5, although the changes are less distinct.

Regarding the frame drop ratio, as illustrated in Figures 7c and 8c, both algorithms demonstrate a declining trend in both resolutions. Specifically, under both algorithms, increase in the failure rate results in a slight decrease in the frame drop ratio for EP scenarios, while the ratio in FP scenarios is less discernible.

In terms of performability, as depicted in Figures 7d and 8d, the observed trend is similar to the trend of throughput. Specifically, in the FP mode, performability exhibits a linear decline with increasing failure rates, regardless of video resolutions. In contrast, the performability of EP appears to remain relatively stable.

In short, failure rates primarily impact service availability, while also influencing service throughput, frame drop ratio and performability. Overall, in our parameter setting, the FP mode demonstrates advantages, achieving higher service availability, increased throughput, greater task handling capacity.

VII. Conclusion

In this study, we proposed SRN models to analyze the performability of UAV-based monitoring systems, particularly focusing on the real-time object detection process and task offloading between drones and fog nodes. To improve the accuracy of model-based performance estimation, we incorporated empirical data from real experiments using an edge device and a PC. Through comprehensive experiments using both YOLOv5 and YOLOv3 as object detection models, we collected the performance data of EP and FP computation modes that were subsequently fed into our proposed SRN models. The numerical results on our models revealed impacts of process failure rates on service availability, throughput, frame drop ratio and overall system performability across different computation modes and video resolutions. When the process failure rate of the edge node is lower than that of the fog node, EP can achieve higher service availability to meet user demands.

This research holds practical relevance across various industries where efficient drone surveillance systems are crucial. Industries such as agriculture, wildlife conservation, security, and infrastructure management stand to benefit significantly from the improved performance and availability offered by our approach. Additionally, integrating empirical data into performance modeling enables more precise analysis and informed decision-making, enhancing the effectiveness of UAV-based monitoring systems. Future work will focus on refining the SRN model, exploring factors like detection accuracy and battery life, and expanding the analysis parameters to create a comprehensive quality analysis and design framework.

Acknowledgment

This work was supported in part by a grant from the Telecommunications Advancement Foundation.

References

[1] V. Kangunde, R. S. Jamisola, and E. K. Theophilus, “A review on drones controlled in real-time,” *International journal of dynamics and control*, pp. 1–15, 2021.

[2] O. Maghazei and T. Netland, “Drones in manufacturing: Exploring opportunities for research and practice,” *Journal of Manufacturing Technology Management*, vol. 31, no. 6, pp. 1237–1259, 2020.

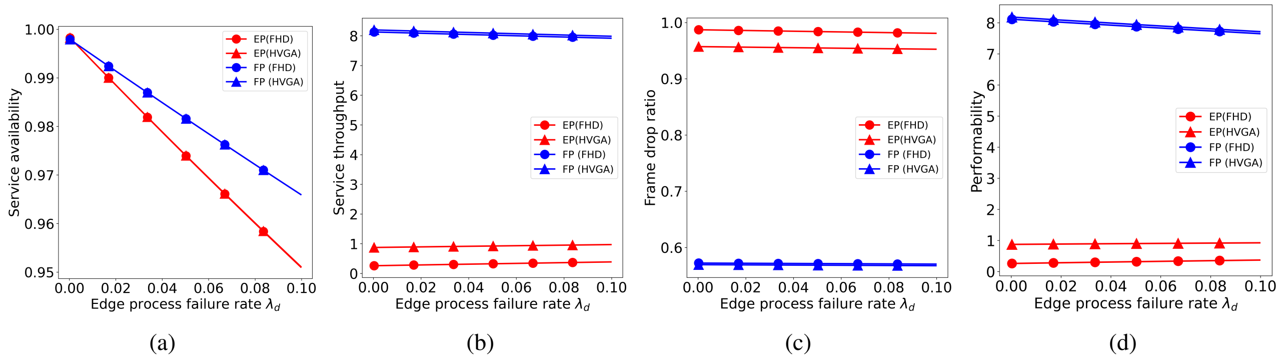


Fig. 7: Sensitivity analysis results by varying λ_d : (a) Service availability; (b) Service throughput; (c) Frame drop ratio; (d) Performability (YOLOv5s)

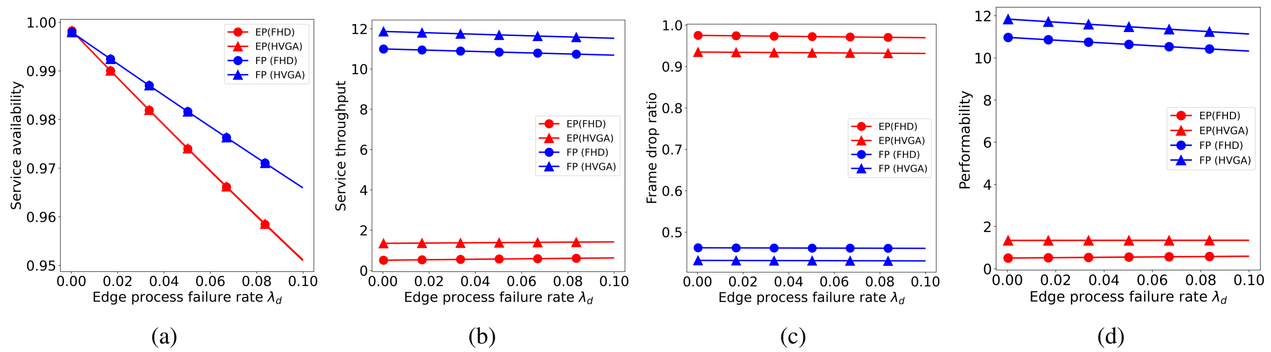


Fig. 8: Sensitivity analysis results by varying λ_d : (a) Service availability; (b) Service throughput; (c) Frame drop ratio; (d) Performability (YOLOv3-tiny)

[3] K. Trivedi, E. Andrade, and F. Machida, "Combining performance and availability analysis in practice," in *Advances in Computers*. Elsevier, 2012, vol. 84, pp. 1–38.

[4] F. Machida and E. Andrade, "Modeling and analysis of deforestation prevention by uncrewed aerial vehicles-based monitoring systems," *Environmental Modelling Software*, vol. 158, p. 105540, 2022.

[5] A. Mukherjee, P. Deb, and D. De, "Natural computing in mobile network optimization," in *Handbook of research on natural computing for optimization problems*. IGI Global, 2016, pp. 382–408.

[6] F. Machida and E. Andrade, "Pa-offload: performability-aware adaptive fog offloading for drone image processing," in *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 2021, pp. 66–73.

[7] Q. Zhang, F. Machida, and E. Andrade, "Performance bottleneck analysis of drone computation offloading to a shared fog node," in *2022 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2022, pp. 216–221.

[8] F. Machida, Q. Zhang, and E. Andrade, "Performability analysis of adaptive drone computation offloading with fog computing," *Future Generation Computer Systems*, vol. 145, pp. 121–135, 2023.

[9] G. Ciardo, A. Blakemore, P. F. Chimento, J. K. Muppala, and K. S. Trivedi, "Automated generation and analysis of markov reward models using stochastic reward nets," pp. 145–191, 1993.

[10] G. Jocher, "Ultralytics yolov5," 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[11] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[12] V. Ajith and K. Jolly, "Hybrid deep learning for object detection in drone imagery: a new metaheuristic based model," *MULTIMEDIA TOOLS AND APPLICATIONS*, vol. 83, no. 3, pp. 8551–8589, 2024.

[13] C. Chen, H. Min, Y. Peng, Y. Yang, and Z. Wang, "An intelligent real-time object detection system on drones," *Applied Sciences*, vol. 12, no. 20, p. 10227, 2022.

[14] W. Shilin, S. Jianli, H. Xiongkui, S. Le, W. Xiaonan, W. Changling, W. Zhichong, and L. Yun, "Performances evaluation of four typical unmanned aerial vehicles used for pesticide application in china," *International Journal of Agricultural and Biological Engineering*, vol. 10, no. 4, pp. 22–31, 2017.

[15] E. Petritoli, F. Leccese, and L. Ciani, "Reliability and maintenance analysis of unmanned aerial vehicles," *Sensors*, vol. 18, no. 9, p. 3171, 2018.

[16] A. Sabino, L. Lima, C. Brito, L. Feitosa, M. Caetano, P. Solis, and F. A. Silva, "Forest fire monitoring system supported by unmanned aerial vehicles and edge computing: A performance evaluation using petri nets," 11 2023.

[17] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, 2023.

[18] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[19] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6105–6114. [Online]. Available: https://proceedings.mlr.press/v97/tan19a.html

[20] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," in *arXiv preprint arXiv:1904.07850*, 2019.

[21] G. Ciardo, J. Muppala, and K. Trivedi, "Spnp: stochastic petri net package," in *Proceedings of the Third International Workshop on Petri Nets and Performance Models, PNPM89*, 1989, pp. 142–151.

[22] R. A. Sahner, K. Trivedi, and A. Puliafito, *Performance and reliability analysis of computer systems: an example-based approach using the SHARPE software package*. Springer Science & Business Media, 2012.

[23] K. S. Trivedi, *Probability & statistics with reliability, queuing and computer science applications*. John Wiley & Sons, 2008.