Intelligent Execution of Computer Vision Tasks in Delay-Constrained UAV-aided Networks

Nancy Varshney*, Corrado Puligheddu †, Carla Fabiana Chiasserini†, Claudio Casetti‡, and Swades De*
* Department of Electrical Engineering, Indian Institute of Technology Delhi, India
†Department of Electronics and Telecommunications, Politecnico di Torino, Italy
‡Department of Control and Computer Engineering, Politecnico di Torino, Italy
E-mail: nancy.varshney@gmail.com, {corrado.puligheddu, carla.chiasserini, claudio.casetti}@polito.it, and swadesd@ee.iitd.ac.in

***Abstract*—Autonomous unmanned aerial vehicles (UAVs) are crucial in critical target tracking and disaster management services. However, challenges arise due to limited channel capacity causing large transmission delays and constraints imposed by UAV batteries when running computationally intensive object detection and tracking algorithms. To address this, we propose intelligent offloading computer vision tasks to a high-computational edge server at millimeter wave (mmWave) frequency. Transmission at mmWave needs large transmission power and is susceptible to blockages that necessitate considering the link quality in the offloading policy. Additionally, the timely processing of frames containing objects of interest is essential for context-based UAV operations to achieve a low frame drop rate. In this work, we present a delayed-reward reinforcement learning framework to determine the offloading policy of computer vision tasks in a delay-constrained environment. This approach considers the importance of the content within frames, which is unknown to the UAV. The objective is to jointly reduce UAV energy consumption and frame drop rates, leveraging statistical information of both the channel and the frame semantics. Through extensive simulations, we demonstrate by considering statistical information of the communication channel and frame semantics, we achieve approximately 45% energy savings compared to the UAV's energy consumption when processing all frames locally and maintaining the drop rate of delay-constrained frames below 5%.**

***Index Terms*—Communication channel, delay, unmanned aerial vehicle (UAV), frame semantics, offloading.**

I. Introduction

Unmanned aerial vehicles (UAVs) have emerged as a prominent technology for surveillance applications, offering the ability to navigate various environments and providing effective monitoring. Moreover, integrating sensing and communication technologies, such as cameras and radars, into UAV systems has allowed advanced computer vision functionalities like object detection, tracking, and obstacle avoidance. In [1], a UAV-based surveillance system with cognitive abilities was proposed, enabling real-time identification of critical situations. However, achieving autonomous UAV operation requires efficient, low-latency object detection capabilities. State-of-the-art video object detection methods, like deep neural networks, are resource-intensive for UAVs, impacting power, bandwidth, and computation, with bulky hardware unsuitable for small drones. Besides sensors, the UAVs are equipped with communication modules for data offloading for the execution of computer vision tasks and exchanging commands with ground control stations (GCS), enabling seamless coordination.

A. Motivation and contribution

Energy-efficient UAV surveillance has been a focus of research in the existing literature, with some works focusing on intelligent UAV navigation and control [2] while others focusing on joint optimization of trajectory and resource allocation for UAV swarm using various reinforcement learning (RL) techniques [3]. Offloading computer vision tasks to a GCS equipped with a high-end processor significantly reduces processing time while maintaining detection accuracy. Additionally, the transmission power is generally lower than the computational power required for object detection. This has been verified at sub-6 GHz in [4], which compares the energy consumption and processing time of running a face recognition algorithm on a video at the UAV versus at GCS.

The study in [5] presented a comprehensive survey of decision-making approaches considering service latency, energy consumption, and processing delay. Similarly, [4] compared the processing delay and energy consumption of a face recognition algorithm over videos of different lengths on a Raspberry Pi unit onboard a UAV and a laptop as the GCS server. The tests were conducted using a UAV integrated with a real-life LTE network and an edge server. [6] proposed an energy-efficient and secure offloading solution that minimizes UAV energy consumption and computational load. Another work [7] examined different modes for image processing and feature extraction at UAV versus edge, considering factors such as network conditions and processing time to determine whether and what to offload. The above-mentioned works highlight a trade-off between power consumption, computational complexity, and processing time when running high-end computer vision algorithms on UAVs. Though using a low-capability server reduces power consumption, it increases processing time, which can be critical for delay-intolerant autonomous UAV operations.

In [8], the authors optimized offloading decisions for computer vision tasks to minimize computation time and UAV energy consumption using a deep RL framework. However, their approach did not consider other time delays, such as

transmission and propagation time of large-sized packets, as well as waiting time in case of unsuccessful frame transfers to the edge. Also, it did not provide insights into the actual frame drop rate, a key metric for assessing system performance.

Moreover, surveillance videos usually consist of a large number of redundant frames with no object of importance. We denote such frames as *non-critical* frames and the frame having the object of interest as *critical* frames. Critical frames usually form a small fraction of the total video, but their timely detection (equivalent to a low drop rate) is crucial for delay-constrained autonomous UAV operations, especially in surveillance scenarios. To this end, various studies explored different schemes for offloading execution of computer vision tasks; the majority primarily focused on computational energy consumption, processing complexity, and processing time as the key metrics. Only a few studies considered communication channel conditions (e.g., [8]), and none accounted for the criticality of data within the frames (or frame semantics) when determining optimal offloading policy.

While employing millimeter-wave (mmWave) technology may overcome the bandwidth insufficiency for transmitting frames with high data bits, in critical scenarios where real-time object detection is essential, it is still advantageous to process the delay-constrained critical frames locally at the UAV when communication channel conditions are poor as mmWave channel is highly susceptible to blockages at the cost of high power consumption at UAV. Thus, there is a trade-off between the frame drop rate and UAV power consumption. Besides, offloading and execution of computer vision tasks based on the semantics of the frame necessitate running the object detection algorithm at the UAV to know if the frame is critical or not, which again consumes energy.

Therefore, in this work, we propose an RL framework for determining an offloading policy for a camera-equipped UAV engaged in surveillance within a designated area at mmWave range, relying on statistical knowledge of the communication channel and frame semantics to make intelligent decisions. The overall contributions of this work are as follows.

1) We propose a delayed-reward State-Action-Reward-State-Action (SARSA) algorithm-based RL framework for designing the optimal offloading decision policy with the aim of reducing the UAV's energy consumption under the constraint of frame drop rates. The future channel transmission condition and frame semantics are modeled as a two-state Markov Decision Process (MDP).
2) We present experimental results of the mean frame execution time and mean energy consumption of running object detection algorithms on the low-power server at UAV as well as high power server at the GCS. These values are later used for RL training.
3) Finally, through extensive simulations, we compare the performance of our proposed framework against two benchmark scenarios. The results demonstrate that leveraging statistical knowledge of both the communication channel and frame semantics significantly enhances the performance of UAV-aided networks by keeping the drop rate of delay-constrained critical frames below 5%.

II. System Model

Consider a camera-equipped UAV following a route R with a time-varying communication link to a GCS for information exchange, as shown in Fig. 1(a). The camera captures a frame every $\Delta t = 1/\text{FPS}$ seconds, where FPS stands for frames per second. Time is divided into slots, with each slot's duration equal to the channel coherence time T_{Coh}. In each slot, the communication link to the GCS can be either good (i.e., perfect data transmission) or bad (i.e., complete link blockage). We model the channel over slots using a two-state MDP and channel transition matrix:

$$P_{Ch} = \begin{bmatrix} p & 1-p \\ 1-q & q \end{bmatrix} \tag{1}$$

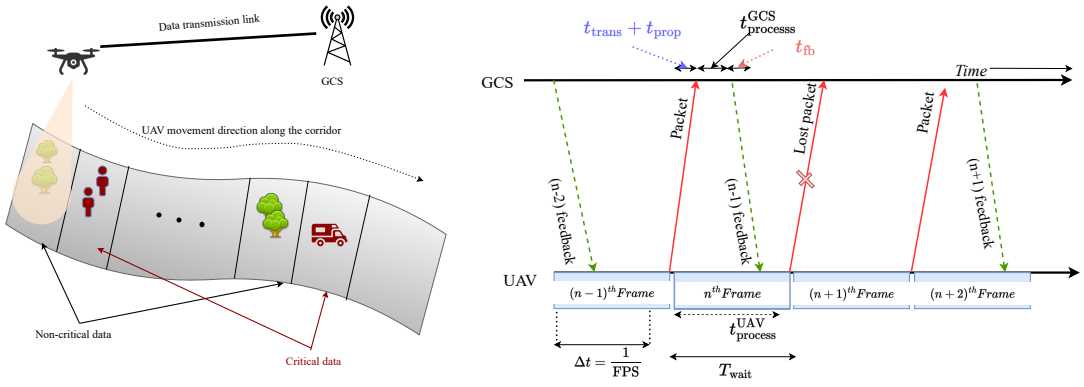
where p, $1-p$, $1-q$, and q denote channel transition probabilities from good to good, good to bad, bad to good, and bad to bad states, respectively. Additionally, the frame data packet transmission occurs over one or more time slots, depending on the data packet size and available channel bandwidth. A frame's complete execution occurs when the channel remains good throughout its execution time at the GCS. Since the video semantics do not change abruptly, we model the frame states using a two-state MDP process, with states given by critical and non-critical, and transition probability:

$$P_F = \begin{bmatrix} v & 1-v \\ 1-w & w \end{bmatrix}. \tag{2}$$

Here, v, $1-v$, $1-w$, and w denote frame transition probabilities from non-critical to non-critical, non-critical to critical, critical to non-critical, and critical to critical states, respectively. Values of $\{p, q\}$ and $\{v, w\}$ are estimated during RL training period based on information obtained from delayed feedbacks.

When n-th frame is processed locally at the UAV, knowledge of the frame's criticality status is obtained after its processing time $t^{\text{UAV}}_{\text{process}}$ (see Fig. 1(b)), with $n = \{1, \ldots, N_F\}$, N_F being the total number of frames over an episode of UAV surveillance. In contrast, when the frame packet is successfully received at the GCS, the UAV anticipates receiving feedback within a duration of T_{wait} after making the decision at time t. This feedback includes criticality status information of the frame and an acknowledgment of its reception. The feedback waiting time T_{wait} depends on following delays (Fig. 1(b)): transmission time delay t_{trans}, propagation time delay t_{prop}, execution time delay at GCS $t^{\text{GCS}}_{\text{process}}$, feedback delay t_{fb}, and buffer delay t_b. Let BW be the bandwidth of the UAV-GCS communication link and B be the mean number of data bits per packet of a frame, then $t_{\text{trans}}{=}B/BW$ and $t_{\text{prop}}{=}d/c$, where d is the 3-dimensional distance of UAV-GCS link and c is the speed of light. Similarly, let B_f be the number of bits in the feedback packet from GCS to UAV, then $t_{\text{fb}}{=}B_f/BW + d/c$. Thus, the total computer vision task execution time is

$$t_{\text{exec}} = \begin{cases} t^{\text{GCS}}_{\text{process}} + 2t_{\text{prop}} + t_{\text{trans}} + t_{\text{fb}}; & \text{executed at GCS} \\ t^{\text{UAV}}_{\text{process}}; & \text{executed at UAV.} \end{cases}$$



(a) (b)

Figure 1: (a) System model of UAV-assisted surveillance over route R. (b) Time diagram of frame packet transmission and processing.

III. Delayed-reward SARSA Framework

RL techniques use experience-based learning and have the advantage of requiring fewer training samples and achieving fast convergence compared to deep learning models [9]. SARSA is particularly well-suited when compared to deep learning models for the small action set in offloading-type problems. In contrast to Q-learning [10], which is commonly used for off-policy RL and is particularly effective for episodic tasks, SARSA exhibits low per-sample variance. Consequently, we employ SARSA to determine the optimal offloading policy, and its corresponding framework is described next.

The on-policy model free SARSA is described by a 4- tuples $\langle S, A, T, R\rangle$, where S is the set of all states, A is the set of all actions, $T(s_{n+1}, a_n, s_n)$=$P(s_{n+1}|s_n, a_n)$ is the transition function, and R is the set of all rewards. The reward function is r_n=$r(a_n, s_n)$. For discount factor $0 \leq \gamma \leq 1$, The objective of the agent is to determine an optimal policy π^*=$P(a_n|s_n)$ that maximizes the expected discounted return

$$R_n = r_n + \gamma r_{n+1} + \gamma^2 r_{n+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{n+k}. \quad (3)$$

Further, the UAV receives the reward r_n on nth frame after a time interval $\Delta t \leq T_{\text{wait}}$. Therefore, we employ delayed reward SARSA, which maintains two copies of the Q-table: the current and the target copy. During training, the agent interacts with the environment, observing the current state, acting according to a policy using the target Q-table, and receiving rewards. The current Q-table is updated as soon as a delayed reward is received. On the other hand, the target Q-table is updated periodically to align with the current network's parameters. In the proposed delayed reward SARSA, the target network is used to estimate the maximum Q-value for the next state. Using a target network, the Q-values in the Q-table can be updated based on a more stable and less volatile estimate of the maximum Q-value of the next state, which improves the learning stability and convergence of the Q-table method.

A. State space

We model states in the RL environment based on communication channel conditions and frame semantics. We assume that the UAV continuously monitors the communication control signals. As described in Sec. II, the communication channel is modeled over the entire allowable duration of a frame execution time, considered as a single unit. Therefore, at the n-th epoch, the agent observes either of the two channel states $s'_n \in S_1 \triangleq$ {good, bad}. To successfully transmit a frame packet to GCS, the UAV requires a minimum of N continuous good time slots, out of N'=$\lceil T_{\text{wait}}/T_{Coh} \rceil$ slots, to complete the transmission, execution, and obtain feedback from the GCS. The value of N is computed as:

$$N = \left\lceil \frac{t_{\text{trans}} + 2t_{\text{prop}} + t^{\text{GCS}}_{\text{process}} + t_{\text{fb}}}{T_{Coh}} \right\rceil. \quad (4)$$

It is important to note that $NT_{Coh} \leq T_{\text{wait}}$. Thus, at the n-th epoch, if in the last N' slots the agent observes that at least N continuous good slots were present, then s'_n=good channel; otherwise bad channel.

We model the RL environment based on frame semantics as well. Therefore, the frame captured at the n-th epoch can have either of the two frame states $s''_n \in S_2 \triangleq$ {critical, non-critical}. If the latest feedback available at the UAV contains the actual frame semantics state, i.e., either critical or non-critical state for $(n-m)^{th}$ frame, then the state of the frame to be processed at epoch n is predicted using MDP as $P(s''_n|s''_{n-m})$=P_F^{n-m}.

Subsequently, there are four states $s_n \in S_1 \times S_2$ as listed in Tab. I. Depending on the state, the agent takes action $a_n \in A$ and receives a delayed reward r_n.

B. Action space

We consider two possible actions $a_n \in A$ at UAV for the n-th frame. The first one, a_n=1, involves offloading the computer vision task to the GCS and is ideally taken when the communication channel is good, irrespective of the frame semantics. This requires less energy to transmit data with $t_{\text{exec}} \ll t^{\text{UAV}}_{\text{process}}$.

Table I: State description

State	Description
$s_n = 1$	{Good channel, Non-Critical frame}
$s_n = 2$	{Good channel, Critical frame}
$s_n = 3$	{Bad channel, Non-Critical frame}
$s_n = 4$	{Bad channel, Critical frame}

The second action, a_n=2, entails processing the frame locally at UAV and is ideally chosen when the communication channel is in a bad state and the frame drop rates cross the threshold values. Even in the latter case, the UAV can opt for action a_n=1 to conserve UAV power consumption if the frame drop rate is below the threshold. However, the state s_n describes the previous channel state and the actual channel during the transmission of the n-th frame follows the true channel statistics. Therefore, experiencing-based learning can lead to suboptimal actions. Moreover, SARSA employs an ϵ-greedy action selection policy [11], i.e., it selects the best action with a probability ϵ to maximize the cumulative reward over a finite time horizon through exploration. In our implementation, we set ϵ to 0.2 and use an ϵ-decay factor of 0.995.

C. Rewards

The reward function is designed based on the remaining UAV energy E_{rem}, while also considering constraints D_{th}^{C} and D_{th}^{NC}, respectively, on the critical frames drop rate D_n^C and non-critical frames drop rate D_n^{NC}. Thus, the optimization problem to determine the optimal policy is expressed as:

$$\begin{aligned} \max_{\pi} \quad & f(E_{\text{rem}}) \\ \text{s.t.} \quad & C1: 0 \leq E_{\text{rem}} \leq E_{\max} \\ & C2: D_n^C \leq D_{th}^C; C3: D_n^C \leq D_{th}^{NC},\ \forall n. \end{aligned} \tag{5}$$

The remaining UAV energy $E_{\text{Rem},n}$ at epoch n, after taking action a_n, is updated as follows

$$E_{\text{Rem},n} = \begin{cases} E_{\text{Rem},n-1} - t_{\text{trans}} P_t; & a_n = 1 \\ E_{\text{Rem},n-1} - t_{\text{process}}^{\text{UAV}} P_{\text{process}}^{\text{UAV}}; & a_n = 2. \end{cases} \tag{6}$$

Here, $P_{\text{process}}^{\text{UAV}}$ represents the frame processing power at the UAV, and P_t is the UAV transmission power. The frame drop rate at epoch n is updated as

$$D_n^{C/NC} = \frac{\text{No. of critical/non-critical packets dropped}}{\text{Total delayed feedbacks available}} \times 100\%\,.$$

To solve (5), we use the reward shaping technique to incorporate the constraints directly. The reward function is appropriately shaped to encourage the agent to take actions that maximize rewards and adhere to the constraints. The total reward over all the N_F frames of a video is defined as

$$\text{Reward} = \frac{1}{N_F} \sum_{n=1}^{N_F} \eta_1 r_{n,1} + \eta_2 r_{n,2} + \eta_3 r_{n,3}\,. \tag{7}$$

In (7), $r_{n,1}$ corresponds to the reward/penalty for the judicious use of remaining UAV energy and is given by:

$$r_{n,1} = \frac{2}{(1 + e^{b_1 \frac{E_{\text{Rem},n}}{E_{\max}}})} \in [0, 1] \tag{8}$$

Reward $r_{n,1}$, with hyperparameter b_1, appropriately incorporates a sigmoidal shape that sensitively responds to UAV energy variations bounded within $[0, 2]$. Given the frame drop rate D_n and drop rate threshold D_{th} at the n-th epoch, let

$$z_n(D_n, D_{th}) = \begin{cases} \left(\frac{E_{\text{rem},n}}{E_{\max}}\right)^{c_1} \left(1 - \frac{2}{\left(1+e^{b_2 \frac{D_n - D_{th}}{D_{th}}}\right)}\right); & D_{th} \neq 0 \\ c_2 \left(\frac{E_{\text{rem},n}}{E_{\max}}\right)^{c_1}; & D_{th} = 0\,. \end{cases}$$

The amplitude of $z_n \in [-1, 1]$ varies as a function of the remaining UAV energy controlled by hyperparameter c_1, while the hyperparameter b_2 controls the shape of z_n around the threshold value D_{th}. Then the reward functions corresponding to the critical frame drop constraint and non-critical frame drop constraint in (7), respectively, are defined as

$$r_{n,2} = z_n(D_n^C, D_{th}^C),\ r_{n,3} = z_n(D_n^{NC}, D_{th}^{NC}). \tag{9}$$

At an epoch, either the frame is critical or non-critical. The value of indicator variables η_1, η_2, and η_3 used for designing the reward system are provided in Tab. II.

D. Q-value update

To update the Q-value, we use expected SARSA, which is a variation of SARSA exploiting the statistical knowledge of communication channels to prevent stochasticity in the policy from further increasing variance. The Q-value for each state-action pair (s_n, a_n) is updated using the rule [11]

$$\begin{aligned} Q(s_n, a_n) = & Q(s_n, a_n) + \alpha[r_n + \\ & \gamma \sum_{s_{n+1}} P(s_{n+1}|s_n) Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)]. \end{aligned} \tag{10}$$

The state $s_{n+1} \in S$ represents the channel condition and frame semantics for the next epoch and is predicted using the joint channel and frame semantics MDP. This information is integrated into the SARSA algorithm to minimize variations in estimating the policy. To simplify the mathematical analysis and predict the availability of a minimum of N consecutive good time slots, we calculate the channel transition probability for the transmission time of the n-th frame by considering the

Table II: State-Action-Reward design for proposed SARSA

{Channel during transmission of n-th frame, n-th frame semantics}	Action a_n	Frame drop rate conditions	Reward indicators
{Good, Non-critical}	a_n=1	$D_n^{NC} \leq D_{th}^{NC}$	$\eta_1 = +1, \eta_2 = 0, \eta_3 = +1$
		$D_n^{NC} > D_{th}^{NC}$	$\eta_1 = +1, \eta_2 = 0, \eta_3 = -1$
	$a_n = 2$	$D_n^{NC} \leq D_{th}^{NC}$	$\eta_1 = -1, \eta_2 = 0, \eta_3 = +1$
		$D_n^{NC} > D_{th}^{NC}$	$\eta_1 = -1, \eta_2 = 0, \eta_3 = -1$
{Good, Critical}	$a_n = 1$	$D_n^C \leq D_{th}^C$	$\eta_1 = +1, \eta_2 = +1, \eta_3 = 0$
		$D_n^C > D_{th}^C$	$\eta_1 = +1, \eta_2 = -1, \eta_3 = 0$
	$a_n = 2$	$D_n^C \leq D_{th}^C$	$\eta_1 = -1, \eta_2 = +1, \eta_3 = 0$
		$D_n^C > D_{th}^C$	$\eta_1 = -1, \eta_2 = -1, \eta_3 = 0$
{Bad, Non-critical}	$a_n = 1$	$D_n^{NC} \leq D_{th}^{NC}$	$\eta_1 = +1, \eta_2 = 0, \eta_3 = -1$
		$D_n^{NC} > D_{th}^{NC}$	$\eta_1 = -1, \eta_2 = 0, \eta_3 = +1$
	$a_n = 2$	$D_n^{NC} \leq D_{th}^{NC}$	$\eta_1 = -1, \eta_2 = 0, \eta_3 = +1$
		$D_n^{NC} > D_{th}^{NC}$	$\eta_1 = +1, \eta_2 = 0, \eta_3 = -1$
{Bad, Critical}	$a_n = 1$	$D_n^C \leq D_{th}^C$	$\eta_1 = +1, \eta_2 = -1, \eta_3 = 0$
		$D_n^C > D_{th}^C$	$\eta_1 = -1, \eta_2 = +1, \eta_3 = 0$
	$a_n = 2$	$D_n^C \leq D_{th}^C$	$\eta_1 = -1, \eta_2 = +1, \eta_3 = 0$
		$D_n^C > D_{th}^C$	$\eta_1 = +1, \eta_2 = -1, \eta_3 = 0$

probability of the next N time slots using the MDP model for the channel. Hence, the state transition probability is

$$P(s'_{n+1}|s'_n) = P_{Ch}^N \,. \tag{11}$$

Further, the considered system forms an uncontrolled MDP where the state transition probabilities do not depend on the action taken [12]. Thus, the overall state transition matrix is

$$P(s_{n+1}|s_n) = P_{Ch}^N \otimes P_F. \tag{12}$$

IV. Results and Discussions

In this section, we first present the experimental values of mean power consumption and mean processing time of execution of computer vision tasks on servers with different computational capacities. Thereafter, we provide the results for the proposed delayed-reward SARSA RL framework that optimize the offloading policy.

A. Experimental results: Mean processing time and mean power consumption

Initially, we examine the mean power consumption and mean processing time values for the execution of the single object tracking (SOT) algorithm and video object detection (VOD) algorithm on the UAV and GCS server at different values of FPS. The specification of the SOT and VOD algorithm, along with the training set used, are given in Tab. III. The UAV server and GCS server details are given in Tab. IV.

Fig. 2(a) shows that using a powerful server at the GCS reduces the processing time of both the SOT and VOD algorithms compared to the local processing of the frames on a small server onboard the UAV. This improvement comes at the cost of higher computational power, as shown in Fig. 2(b), while maintaining an accuracy of 70.3%.

Algorithm 1 Delayed-reward SARSA algorithm

Initialize current and target Q-tables, α, γ, Δ_{target}, P_F, P_{Ch}
$S=\emptyset$, $S_{\text{next}}=\emptyset$, $A=\emptyset$, $A_{\text{next}}=\emptyset$
Initialize state s_0 and $s''_{\tilde{n}}$; $\{S\}\leftarrow\{S\}\cup s_0$
Sample a_0 form ϵ-greedy policy $\pi(s_n, a_n)$ and $\{A\}\leftarrow\{A\}\cup a_0$
for each episode **do**
  for each frame or step **do**
    Find remaining UAV energy E_{rem}
    Update P_{Ch}
    Predict $\bar{s}_{n+1}$ using (12)
    $\{S_{\text{next}}\}\leftarrow\{S_{\text{next}}\}\cup\bar{s}_{n+1}$
    Sample $\bar{a}_{n+1}$ from policy $\pi(\bar{s}_{n+1}, \bar{a}_{n+1})$
    using target Q-table
    $\{A_{\text{next}}\}\leftarrow\{A_{\text{next}}\}\cup\bar{a}_{n+1}$
    if delayed reward present **then**
      Get the reward for frame $\tilde{n}$, $\tilde{n}\leq n$
      Update P_F
      Update current Q-table using (10)
      $\{S\}\leftarrow\{S\}\backslash s_{\tilde{n}}$; $\{A\}\leftarrow\{A\}\backslash a_{\tilde{n}}$
      $\{S_{\text{next}}\}\leftarrow\{S_{\text{next}}\}\backslash a_{\tilde{n}+1}$; $\{A_{\text{next}}\}\leftarrow\{A_{\text{next}}\}\backslash a_{\tilde{n}+1}$
      $s''_{\tilde{n}}\leftarrow$ true $\tilde{n}^{th}$ frame semantics
    end
    if remainder of (step/Δ_{target}) == 0 **then**
      Replace: target Q-table $\leftarrow$ current Q-table
    end
    Update s'_{n+1} with past channel information
    Predict s''_{n+1} using P_F and $s''_{\tilde{n}}$
    Determine s_{n+1}; $\{S\}\leftarrow\{S\}\cup s_{n+1}$
    Sample a_{n+1} from policy $\pi(s_{n+1}, a_{n+1})$ using target Q-table
    $\{A\}\leftarrow\{A\}\cup a_{n+1}$
    Update α
  end
end

Table III: Object detection and tracking algorithm specifications

Algorithm	VOD	SOT
Model	SiamRPN++	Deep feature flow
Backbone	R-50	R-50-DC5
Training dataset	ImageNET VID	LaSOT
Accuracy	50.4 Success	70.3 box AP@50

Table IV: UAV and GCS server specifications

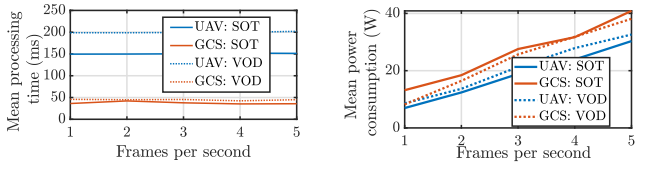
Server	Server 1	Server 2
Class	UAV	GCS
CPU	Intel i7-7700HQ	2× AMD EPYC 7601
CPU TDP (W)	45	2 × 180
GPU	Nvidia MX150	Nvidia GV100
GPU TDP (W)	25	250

B. Simulated results: SARSA

We now present the performance of the SARSA algorithm under the following settings: carrier frequency=28 GHz, BW=400 MHz, B=10 MB, UAV transmit power P_t=1 W, T_{Coh}=10 ms, FPS=5, N_F = 1000, T_{wait}=200 ms. As a result, t_{trans}=25 ms; also, t_{prop}=0.0003 ms. Assuming feedback packet of 1 KB, t_{fb}=0.0025 ms and N=8. From the results in Sec. IV-A, at FPS=5, we have $t_{\text{process}}^{\text{UAV}}$=151.3 ms, $t_{\text{process}}^{\text{GCS}}$=35.72 ms, and the mean power required for running the SOT algorithm at the UAV and GCS are 30.34 W and 40.89 W, respectively. Also, p=0.9, q=0.8, v=0.85, and w=0.85. The RL simulation parameters are γ=0.4, D_{th}^{C}=0%, D_{th}^{NC}=5%, b_1=3, b_2=3, c_1=0.01, c_2=0.8, Δ_{target}=2 s, and α=0.2. Further, we compare our proposed 4-states RL framework with the following two benchmarks RL frameworks:

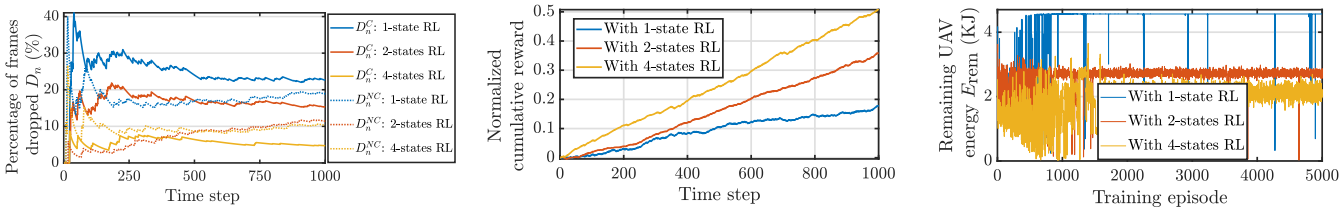
- *1-state RL*: In this benchmark, there is only 1 RL state, and the agent can choose any action $a_n \in A$;
- *2-states RL*: In this benchmark, there are 2 RL states $s' \in S'$, which correspond to the mmWave communication channel. The agent can take any action $a_n \in A$.

Fig. 3(a) highlights the impact of leveraging frame semantics on the performance of D_n^C and D_n^{NC} in approaching threshold values, surpassing the results obtained solely through the utilization of channel statistical information to train the UAV. In contrast, employing neither channel nor frame semantics leads to the most severe packet loss, as it overfits the model. It is notable that the values of D_n^C and D_n^{NC} do not



(a) Mean processing time t_{process} (b) Mean power consumption P_{process}

Figure 2: Mean processing time t_{process} (a) and mean power consumption P_{process} (b) of the SOT and VOD algorithms, when implemented on UAV server and GCS server.



(a) Frame Drop rates $D_n^{C/NC}$ (b) Normalized cumulative reward (c) Remaining UAV energy E_{rem}

Figure 3: (a) Convergence of D_n^C and D_n^{NC} over time steps of last training episode, (b) convergence of normalized cumulative reward over time steps of last training episode, and (c) convergence of remaining UAV energy over 5000 training episodes, with the three RL frameworks.

satisfy the required threshold constraints since, in our proposed RL framework, these constraints have been designed using reward functions that either reward or penalize the system based on whether the constraints are met or not.

In case no intelligence is used at UAV, processing all frames locally on the UAV results in the maximum energy expenditure of E_{max}=4.9505 KJ, whereas offloading all frames to the GCS incurs a mere 0.025 KJ of UAV energy, although at the cost of significantly higher frame drop rates.

Fig. 3(b) compares the cumulative rewards achieved, while Fig. 3(c) analyzes the remaining UAV energy across three RL frameworks. The proposed 4-states RL framework stands out as it achieves the lowest critical frame drop rate of 5% while saving approximately 45% of UAV energy and achieving highest reward. The 2-states RL framework achieves approaximately 60% reduction in UAV energy consumption but comes with a higher critical frame drop rate of 10%. In contrast, the 1-state RL framework achieves 99.5% energy savings by offloading all frames to the GCS but this comes at the cost of a substantial 25% drop in critical frames.

V. Conclusions

The quality and reliability of the channel directly impact data transmission efficiency, making it essential to consider the communication channel condition when deciding whether to offload frames to the GCS. The existing literature lacks attention to communication channel constraints and the criticality of data content to achieve low frame drop rates. To optimize power usage and maximize the UAV's operation, frames can be offloaded to the GCS if the channel is good allowing the UAV to allocate its resources to process critical frames during unstable channel conditions. To address these gaps, this study designed a UAV-aided monitoring mechanism that considers UAV energy limitations, frame drop rate constraints for timely object detection, and communication channel constraints. Moreover, using the delayed-reward SARSA framework, the UAV can strike a balance between transmitting all frames to the GCS for processing and processing all frames locally with optimized power usage. UAV energy savings of up to 45% compared to locally processing all the frames can be achieved while maintaining low frame drop rates, particularly of delay-constrained critical frames below 5%. Exploration of the optimization of reward functions based on parameters beyond the UAV's remaining energy to enhance system performance will be undertaken as part of future work.

VI. Acknowledgement

This work was supported in part by the Indian Ministry of Science and Technology and the Italian Ministry of Foreign Affairs through the Executive Programme for Scientific and Technological Cooperation (project no. IN22MO05); in part by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART"); in part by the Science and Engineering Board, Dept. of Science and Technology (DST), Government of India, under Grant CRG/2019/002293, and in part by the Indian National Academy of Engineering (INAE) through the Abdul Kalam Technology Innovation National Fellowship.

References

[1] D. Cavaliere, S. Senatore, and V. Loia, "Proactive uavs for cognitive contextual awareness," *IEEE Systems Journal*, vol. 13, no. 3, pp. 3568–3579, 2018.

[2] M. J. Er, C. Ma, T. Liu, and H. Gong, "Intelligent motion control of unmanned surface vehicles: A critical review," *Ocean Engineering*, vol. 280, p. 114562, 2023.

[3] F. Koohifar, A. Kumbhar, and I. Guvenc, "Receding horizon multi-UAV cooperative tracking of moving RF source," *IEEE Communications Letters*, vol. 21, no. 6, pp. 1433–1436, 2016.

[4] N. H. Motlagh, M. Bagaa, and T. Taleb, "UAV-based IoT platform: A crowd surveillance use case," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 128–134, 2017.

[5] S. A. Huda and S. Moh, "Survey on computation offloading in UAV-Enabled mobile edge computing," *Journal of Network and Computer Applications*, vol. 201, p. 103341, 2022.

[6] T. Bai, J. Wang, Y. Ren, and L. Hanzo, "Energy-efficient computation offloading for secure UAV-edge-computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 6074–6087, 2019.

[7] S. Hayat, R. Jung, H. Hellwagner, C. Bettstetter, D. Emini, and D. Schnieders, "Edge computing in 5G for drone navigation: What to offload?" *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2571–2578, 2021.

[8] D. Callegaro and M. Levorato, "Optimal edge computing for infrastructure-assisted UAV systems," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1782–1792, 2021.

[9] S. Tripathi, C. Puligheddu, C. F. Chiasserini, and F. Mungari, "A context-aware radio resource management in heterogeneous virtual RANs," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 1, pp. 321–334, 2021.

[10] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE access*, vol. 7, pp. 133 653–133 667, 2019.

[11] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[12] J. P. Epperlein, R. Overko, S. Zhuk, C. King, D. Bouneffouf, A. Cullen, and R. Shorten, "Reinforcement learning with algorithms from probabilistic structure estimation," *Automatica*, vol. 144, p. 110483, 2022.