ReMO: Adaptive Region-Based Offloading for Collaborative Edge Video Analytics

Yanni Xing[1], Yu-E Sun[1*], Ning Chen[2,3*], Hao Pan[2], Sheng Zhang[3], Jie Wu[4]
[1]School of Rail Transportation, Soochow University, Suzhou, China
[2]School of Computer Science and Technology, Soochow University, Suzhou, China
[3]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[4]Center for Networked Computing, Temple University, Philadelphia, USA
{ynxing02, hpanpanhao}@stu.suda.edu.cn, {sunye12, ningc}@suda.edu.cn, sheng@nju.edu.cn, jiewu@temple.edu

2025 IEEE 31th International Conference on Parallel and Distributed Systems (ICPADS) | 979-8-3315-4901-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICPADS67057.2025.11323003

***Abstract*—With the proliferation of edge computing and the Internet of Things (IoT), inference-driven intelligent cameras are increasingly deployed for resource-efficient and privacy-preserving processing. In real-world scenarios, such as traffic surveillance, cameras deployed at different locations (e.g., intersections or corners) experience imbalanced inference workloads, leading to latency bottlenecks and resource underutilization. To address this, we propose ReMO, an adaptive framework for collaborative video analytics. Unlike full-frame offloading, ReMO divides video frames into regions to reduce data transmission and enable fine-grained load balancing. It consists of two key components: a Region Generator that analyzes scene features to identify regions with varying detection needs, and a Region Scheduler that formulates scheduling as an integer nonlinear problem, solved via an adaptive online algorithm based on Lyapunov optimization and Markov approximation. Experimental results demonstrate that ReMO effectively reduces latency by 9.3–26.3% while maintaining high accuracy, outperforming baseline strategies.**

***Index Terms*—Edge Computing, Collaborative Inference, Adaptive Offloading, Lyapunov Optimization**

I. Introduction

Deep Neural Networks (DNNs) serve as the foundation for numerous cutting-edge AI applications, including object detection [1], semantic segmentation [2], text generation [3], and gesture recognition [4]. Benefited from the widely studied *Mobile Edge Computing (MEC)* paradigm, DNN inference tasks can be processed closer to the data source, thereby reducing latency and improving privacy. Coupled with the rapid proliferation of the *Internet of Things (IoT)* and advances in *wireless sensing* [5], edge intelligence has become a key enabler for decentralized AI. Among these applications, video analytics has emerged as a primary use case [6], [7], where inference workloads are increasingly handled by edge devices such as smart cameras.

However, real-world deployments (Fig. 1) show significant spatial and temporal workload imbalance across cameras. Overloaded cameras at busy intersections(camera 1, 3) suffer from increased inference latency caused by queuing delays, while underutilized cameras in quieter areas (camera 2) fail to make efficient use of their available resources.

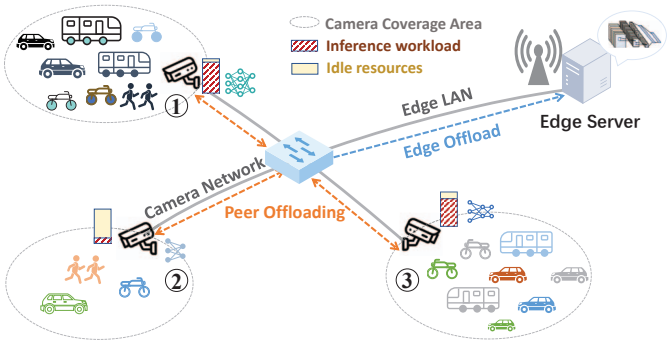


Fig. 1: Workload imbalance in traffic surveillance scenario

A natural and promising approach to addressing this imbalance is to redistribute the inference workload by offloading tasks from overburdened devices to underutilized peer edge nodes. However, devising an optimal workload redistribution strategy is inherently complex, posing three key challenges:

Firstly, offloading a large volume of data imposes significant costs in terms of device capacity and network transmission overhead. Original video streams typically feature high resolutions (e.g., 1080p), demanding substantial computational power and bandwidth to support real-time inference and data transfer. This requirement is at odds with the inherent constraints of edge devices, which often suffer from limited computational and network resources. The challenge becomes even more pronounced when dealing with ultra-high-definition video (e.g., 4K) or when network reliability between edge devices is suboptimal. A seemingly straightforward solution is to partition video frames into multiple equal-sized tiles [8] and distribute them across cameras for processing. However, this approach introduces excessive tile boundaries, necessitating massive cross-tile processing.

Secondly, each camera's inference workload exhibits inherent spatiotemporal variability. Spatially, object distribution within a frame is uneven, with certain regions consistently covering more objects than others. Temporally, these workload patterns remain dynamic and unpredictable due to fluctuating

*Yu-E Sun (sunye12@suda.edu.cn) and Ning Chen (ningc@suda.edu.cn) are co-corresponding authors.

This work was supported in part by NSFC (Grant No. 62502331), China Postdoctoral Science Foundation (Grant No. 2025M771496), State Key Lab. for Novel Software Technology of Nanjing University (Grant No. KFKT2025B24), Jiangsu Funding Program for Excellent Postdoctoral Talent.

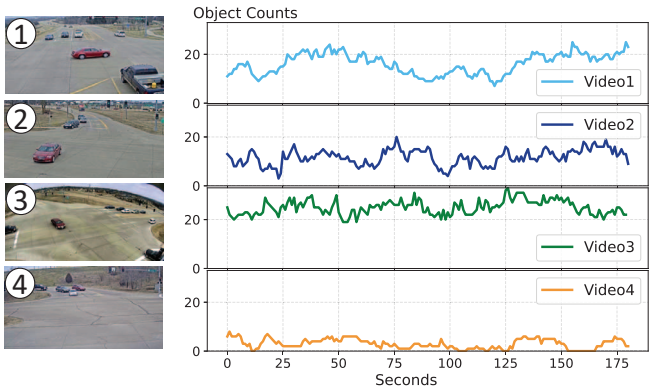
traffic flows influenced by factors such as camera location and time. While the number of detected objects provides a coarse workload estimate, the lack of real-time video analytics precludes accurate characterization of spatiotemporal workload dynamics. This dual uncertainty – in where and when computational demands peak – fundamentally limits the effectiveness of static offloading strategies, often causing persistent resource imbalances. Consequently, achieving an accurate and efficient estimation of spatiotemporal workload is crucial for adaptive resource allocation.

Thirdly, model heterogeneity and network fluctuations necessitate a comprehensive approach for real-time decision-making. Surveillance edge cameras in urban environments typically handle heavier workloads compared to their suburban counterparts, leading to the deployment of diverse CNN architectures. However, more accurate models inevitably incur longer inference latency. This variance in processing delay and detection performance significantly complicates the provision of real-time guarantees. Furthermore, the inherent volatility of edge network bandwidth invalidates static scheduling, especially when transmitting data over congested network links where bandwidth contention leads to substantial and unpredictable transmission delays. These challenges underscore the need for an adaptive scheduling algorithm that can accommodate both model heterogeneity and network variability.

Existing research efforts fall short in addressing these challenges comprehensively. Some studies [8]–[11] have explored region-of-interest (RoI)-based offloading to reduce inference costs by leveraging spatial redundancy within frames. However, these methods rely on time-intensive CNN-based predictions for RoI selection, introducing additional computational overhead. Other works [12], [13] have focused on adaptively balancing computational loads across cameras but do not account for intra-frame spatial redundancy.

To address both spatial redundancy and dynamic workload imbalance, we propose **ReMO**, a region-based offloading framework built upon the widely-used Detection-Based Tracking (DBT) paradigm. ReMO consists of two key components. First, the **Region Generator** partitions frames into fixed regions based on long-term trajectory observations—eschewing CNN-based proposals [11], [14] to minimize overhead. To reflect the spatiotemporal workload, we quantify the detection necessity of each region in every frame using multi-dimensional metrics (e.g., optical flow standard deviation) with adaptive padding subsequently applied to preserve object integrity. Second, the centralized **Region Scheduler** solves an integer nonlinear program in real-time using a Lyapunov-optimized Markov approximation algorithm, dynamically adapting to content variations, model heterogeneity, and bandwidth fluctuations for robust scheduling.

We implement ReMO on commercial off-the-shelf (COTS) hardware and conduct comprehensive evaluations using multi-camera datasets. Results show that ReMO adaptively balances workloads, reducing response latency by 9.3–26.3% while maintaining high accuracy. The implementation is open-sourced at https://github.com/yannixing/ReMO.git. Our major



(a) Videos (b) Variation of object counts

Fig. 2: Workloads vary across different videos

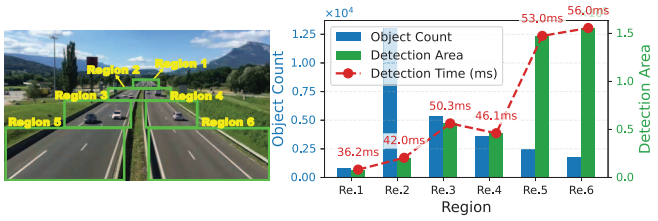
contributions are as follows:

- We propose region-based offloading based on fixed, model-free region partitioning without excessive overhead and boundary handling to mitigate spatial redundancy while enabling fine-grained load balancing.
- We design a multidimensional metric inspired by tracking drift factors to estimate spatiotemporal workload and dynamically trigger region detection per frame.
- We formulate and solve the adaptive scheduling problem through an online algorithm combining Lyapunov optimization and Markov approximation, effectively handling model heterogeneity and network dynamics.
- We conduct extensive experiments demonstrating the superior performance of our approach compared to baselines in terms of accuracy, latency, and load balancing.

II. Motivations

Cross-Camera Workload Imbalance: Traffic cameras [15] are often deployed with complementary fields of view, but their inference workloads can differ due to scene coverage. As shown in Fig. 2b, Videos 1 and 3 monitoring major intersections consistently experience heavier workloads than Videos 2 and 4. These patterns, however, are highly dynamic. Traffic conditions can change abruptly, as shown in Video 2 where the object count surges between 23 and 28 seconds. Moreover, prior work [12] has shown that dense scenes under the DBT framework require more frequent detection to mitigate tracking drift. Such temporal and content-driven fluctuations render static offloading strategies ineffective, highlighting the need for adaptive, content-aware scheduling mechanisms.

Intra-Camera Workload Imbalance: Object density varies spatially due to stable road layouts. Rather than using costly region proposal methods (e.g., ELF [11]), we partition frames into fixed regions derived from historical trajectory patterns (Fig. 3a). As shown by the blue bars in Fig. 3b, some regions consistently contain more objects and require higher computational effort. Notably, due to poor image quality, Region 1 exhibits a lower-than-expected object count, despite covering a high-traffic road area. This intra-frame workload skew mirrors the cross-camera imbalance, as both stem from uneven object



(a) Partitioning example (b) Detection results by region

Fig. 3: Workloads vary across different regions

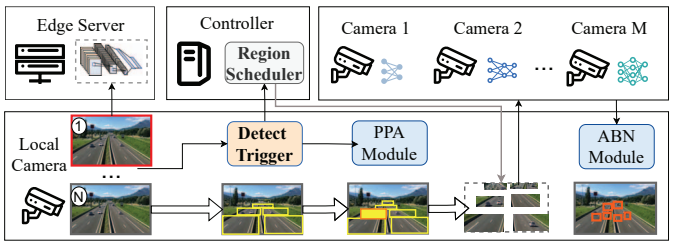


Fig. 4: System architecture of ReMO

distributions shaped by physical scene characteristics and content dynamics.

Benefits of Region-Based Offloading: To assess the cost of region-level inference, we benchmarked the YOLOv8n model on an NVIDIA Jetson Nano B01. As illustrated by the red scatters and green bars in Fig. 3b, inference time increases approximately linearly with region area—larger regions (e.g., Region 5) incur significantly longer delays than smaller ones. Region-based offloading allows selective processing of high-value regions, reducing redundant computation and data transmission. This not only alleviates bandwidth bottlenecks but also enables fine-grained load balancing across collaborative devices, supporting scalable and real-time inference under constrained conditions.

These observations motivate ReMO, our region-based offloading framework that dynamically allocates tasks across devices using spatial-temporal workload patterns.

III. System Architecture

For simplicity, we manually define rectangular regions offline based on motion trajectories and scene layout. The fixed region coordinates are stored locally to enable lightweight and consistent partitioning without runtime computation. Without loss of generality, we segments time into discrete slots and adopts the DBT framework. Since tracker performance depends heavily on the initial detection quality, each camera offloads the first frame of every slot to the edge server for accurate object detection.

Workflow: As illustrated in Fig. 4, ReMO consists of four key modules. First, the local camera offloads the first frame of each video chunk to the edge server. Subsequent frames adopt the DBT method, where object detection is triggered only when lightweight trackers (e.g., optical flow) drift. The **Detect Trigger** is designed to calculate multi-dimensional metrics for predefined regions (without frame cropping) using Python functions, determines the regions requiring detection and then transmits selected region IDs to the Region Scheduler. Second, the Proportional Padding Adjustment (PPA) Module introduces a padding mechanism that enlarges the selected regions ensuring complete bounding boxes for boundary objects. These cropped regions are then packaged into inference tasks and assigned to devices based on scheduling.

Third, the **Region Scheduler** (in a separate controller) aggregates region IDs from all devices and pre-stores region-specific data (e.g., camera-captured area per ID) to aid decision-making. The scheduler dynamically assigns these tasks using an adaptive online algorithm. Finally, each camera, running a DNN model, processes its allocated tasks and returns results to the corresponding cameras. Then, the Area-Based Non-Maximum Suppression (ABN) Module filters bounding boxes, retaining only the largest and most complete detections while discarding partial or redundant ones. This process achieves seamless cross-region object detection fusion.

To streamline the system design, ReMO consolidates the Detect Trigger and PPA Module into a single **Region Generator** (see Section IV). The methodology of **Region Scheduler** is detailed in Section V.

IV. Region Generator Design

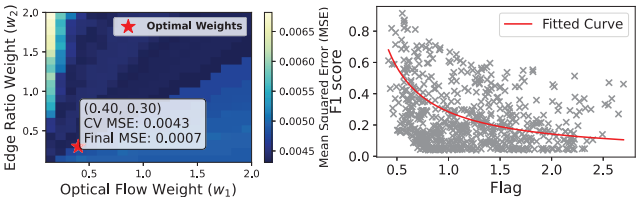
A. Detect Trigger

To assess the necessity of detection in these regions, we analyze tracking failure causes in the DBT framework. The primary issue stems from accumulated drift in long-term optical flow tracking, where estimated positions progressively deviate from ground truth. This drift originates from two sources: (1) tracking failure of existing objects (detectable through optical flow point dynamics), and (2) new object emergence (determined by region-specific properties). We quantify these phenomena through two dedicated metrics.

Optical Flow Standard Deviation: Prior studies [16] have demonstrated that the standard deviation of dense optical flow serves as a reliable indicator of motion inconsistencies, which are strongly correlated with tracking failures. Consequently, a higher optical flow standard deviation suggests potential tracking failure, necessitating region detection. To ensure real-time efficiency, we employ `cv2.DISOpticalFlow_create` to compute dense optical flow. The optical flow standard deviation for region r is denoted as s_r.

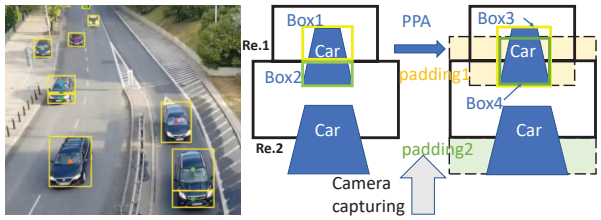
Edge Pixel Ratio: When a new object enters a region, the optical flow mechanism cannot independently establish feature points for tracking. However, new objects significantly alter the distribution of edge pixels. Therefore, monitoring edge pixels serves as an indicator of object entry into a region. Edge pixels are detected using `cv2.Canny`, and the normalized edge pixel ratio for region r is defined as: $e_r = \frac{E_r}{M_r}$, where E_r denotes the total number of edge pixels in region r, and M_r represents the total number of pixels in region r.

Flag: We combine s_r and e_r into a unified score: $\text{Flag}[r] = w_1 \cdot s_r + w_2 \cdot e_r$, where (w_1, w_2) are optimized through grid search with 5-fold cross-validation, minimizing the mean squared error (MSE) between predicted and actual F1-score drops (performing full-frame detection on the first frame per time slot and switching to optical flow for subsequent frames).



(a) Cross-validation seeking (b) Inversely related

Fig. 5: Determining weights



(a) Object split (b) The principle of PPA

Fig. 6: The process of PPA

Fig. 5a shows the weight optimization process for video [17], with the optimal weights $(0.4, 0.3)$ yielding flag values that inversely correlate with F1-scores (Fig. 5b). To enable real-time adaptation, we monitor $\frac{d\text{Flag}}{dt}$ and trigger detection when it exceeds a threshold Q, which is dynamically adjusted based on system latency.

B. PPA Module

Although regions are divided based on motion trajectories to minimize object segmentation, cross-region objects still require handling. As shown in Fig. 6a, some vehicles are detected with partial bounding boxes across two regions.

To address this, we employ the PPA module that adaptively expands region boundaries for complete object detection (Fig. 6b). The padding extends shared boundaries (e.g., Box3), with size determined by a perspective-aware adaptive ratio. Regions closer to the camera receive larger padding (padding 2 $>$ padding 1 in Fig. 6b) as objects appear larger. For boundaries parallel to the camera orientation, we apply the average of adjacent padding ratios. This design ensures the high accuracy of region-based offloading.

V. Region Scheduler Design

This section introduces the *Region Scheduler*, which adaptively balances workloads across cameras by formulating and solving an optimization problem.

A. System models

We consider a system of smart cameras $\mathcal{N} = \{1, 2, ..., N\}$ and one edge server (0). Each camera's video frame is divided into R_i fixed regions, with region area denoted as M_r and full-frame area M_0. Table I summarizes key notations.

Time is partitioned into slots $\mathcal{T} = \{1, 2, ..., T\}$, with each camera capturing F frames per slot. The scheduling matrix X_t^i records the task allocation for camera i in slot t: if $X_t^i[f, r] = j$, region r in frame f is offloaded to camera j; if $X_t^i[f, r] = -1$, optical flow tracking is used locally.

TABLE I: Summary of Notations Used for System Model

Notation	Description
M_r	Area of region r
M_0	Area of frame
R_i	Number of regions for camera i
X_t^i	Scheduling policy of camera i in slot t
$\mathbf{X}_t$	Set of scheduling policies for N cameras
$s_{i,M}$	Inference time for camera i to process an image with an area of M
a_i	Average detection accuracy of camera i
a_0	Average detection accuracy of edge servers
$\phi_{i,t}$	Tracking drift factor for camera i in slot t
$A_{i,t}$	Accuracy of camera i in slot t
A_t	Average accuracy of N cameras in slot t
s_0	Detection delay of edge servers
$T_{i,t}^{det}$	Total detection delay of camera i in slot t
$b_t^{i,j}$	Bandwidth share from camera i to camera j in slot t
$T_{i,t}^{trans}$	Transmission delay of camera i in slot t
$T_{i,t}$	Delay of camera i in slot t
T_t	Maximal delay of all cameras (system delay) in slot t
$K_{i,t}$	Load of camera i in slot t
K_t	System load balancing index in slot t

Accuracy Model: For detected regions, accuracy is $a_{X_t^i[f,r]}$. For tracked regions, accuracy degrades with tracking drift $\phi_{i,t}$ and prior accuracy $A_{i,t}^{f-1,r}$. The frame accuracy is the average of all region accuracies:

$$A_{i,t}^f = \sum\nolimits_r^{R_i} \left(1 - \mathbb{I}[X_t^i[f,r] = -1]\right) \cdot a_{X_t^i[f,r]} + \mathbb{I}[X_t^i[f,r] = -1] \cdot A_{i,t}^{f-1,r} \cdot \phi_{i,t}, \tag{1}$$

where $\mathbb{I}[X_t^i[f, r] = -1]$ indicates whether optical flow tracking is applied to this region.

The per-camera accuracy is $A_{i,t} = \frac{1}{F}\sum_f A_{i,t}^f$, and the average accuracy of N cameras is:

$$A_t = \frac{1}{N}\sum\nolimits_{i\in\mathcal{N}} A_{i,t}. \tag{2}$$

Delay Model: Inference delay of camera i depends on detection area and model speed:

$$T_{i,t}^{det} = \sum\nolimits_{r\in\{r|X_t^j[f,r]=i, j\in\mathcal{N}\}} s_{i,M_r}. \tag{3}$$

Transmission delay $T_{i,t}^{trans}$ includes sending the first frame to the server and subsequent data transfer. For the latter, the sending delay is $T_{i,t}^{send} = \sum_{j\in\mathcal{N}} \frac{\sum_{r\in\{r|X_t^i[f,r]=j\}} \alpha M_r}{b_t^{i,j}}$, where α is a constant representing the ratio between region area and data size. Similarly, the receiving delay is $T_{i,t}^{recv} = \sum_{j\in\mathcal{N}} \frac{\sum_{r\in\{r|X_t^j[f,r]=i\}} \alpha M_r}{b_t^{j,i}}$. Thus, the transmission delay of camera i is

$$T_{i,t}^{trans} = \frac{\alpha M_0}{b_t^{i,0}} + T_{i,t}^{send} + T_{i,t}^{recv}. \tag{4}$$

In conclusion, the total delay of camera i is $T_{i,t} = T_{i,t}^{det} + T_{i,t}^{trans}$. The system delay is

$$T_t = \max(T_{i,t}). \tag{5}$$

System Load Balancing Index: The load balancing among cameras can be quantified by defining each camera's load as the total image area requiring detection. The system load balancing index K_t is the variance of all cameras' loads:

$$K_t = \frac{1}{N}\sum\nolimits_{i\in\mathcal{N}} \left(K_{i,t} - \frac{1}{N}\sum\nolimits_{i\in\mathcal{N}} K_{i,t}\right)^2. \tag{6}$$

B. Problem Formulation

The optimization goal is to maximize utility (accuracy minus imbalance) under long-term delay constraints:

$$\mathcal{P}^G : \max \lim_{T\to+\infty} \frac{1}{T} \sum_{t=0}^{T-1} (A_t - \omega K_t) \tag{7}$$

$$\text{s.t.} \lim_{T\to+\infty} \frac{1}{T} \sum_{t=0}^{T-1} T_t \leq T_{max} \tag{8}$$

$$\forall r, \exists f, X_t^i[:, r] \neq -1, i \in \mathcal{N}, t \in \mathcal{T} \tag{9}$$

where ω is the weighted parameter to regulate the trade-off between accuracy and load balancing. The constraint (8) ensures that the system delay will not exceed the delay threshold T_{max} in the long term. The constraint (9) guarantees that each region is detected at least once per slot.

Solving problem $\mathcal{P}^G$ optimally requires near-future knowledge of network conditions and video content. As an NP-hard problem, $\mathcal{P}^G$ remains difficult even with future information, motivating the need for an online solution.

C. Algorithm design

Due to the long-term constraint, we apply Lyapunov optimization to decompose $\mathcal{P}^G$ into per-slot problems [18].

1) Lyapunov Optimization: A virtual delay queue $q(t)$ tracks delay violations:

$$q(t+1) = \max[q(t) + (T_t - T_{max}), 0]. \tag{10}$$

To describe the queue stability, we define a quadratic Lyapunov function as $L(q(t)) = \frac{1}{2}(q(t))^2$. Consequently, we obtain the Lyapunov drift, i.e.,

$$\Delta(q(t)) = \mathbb{E}\left[L(q(t+1)) - L(q(t)) \mid q(t)\right]. \tag{11}$$

Considering the utility function, we introduce the Lyapunov drift-plus-penalty:

$$\Delta(q(t)) - V \cdot \mathbb{E}[A_t - \omega K_t | q(t)] \tag{12}$$

Typically, the min-drift-plus-penalty algorithm does not directly minimize the drift-plus-penalty of each slot but instead minimizes its upper bound [18]. As a result, we conclude the following lemma:

Lemma 1: For $q(t)$ in any slot the statement holds:

$$\begin{aligned}\Delta(q(t)) - V \cdot \mathbb{E}\left[A_t - \omega K_t \mid q(t)\right] \leq C + \\ q(t) \cdot \mathbb{E}\left[(T_t - T_{\max}) \mid q(t)\right] - V \cdot \mathbb{E}\left[A_t - \omega K_t \mid q(t)\right],\end{aligned} \tag{13}$$

where $C = \frac{1}{2}(T_{\max,t} - T_{\max})^2$ is a constant for all slots and $T_{\max,t} = \max_{t\in\mathcal{T}}\{T_t\}$ denotes the maximum system delay for all slots. The complete proof can be found in [18].

Next, we derive the one-slot optimization problem $\mathcal{P}^L$:

$$\begin{aligned}\mathcal{P}^L : &min\, q(t) \cdot T_t - V \cdot (A_t - \omega K_t) \\ &\text{s.t.}(13), (14).\end{aligned} \tag{14}$$

2) Markov Approximation: It is important to note that $\mathcal{P}^L$ is also an integer nonlinear problem, which, as such, cannot be solved to find the optimal solution in polynomial time. Consequently, we employ the Markov approximation [19] method to construct a Markov Chain for iterative solution. In the iteration process, the algorithm selects the next state based on the change in the objective function value, expecting the objective function value to gradually decrease during the iterations.

Algorithm 1 Algorithm 1: Solve $\mathcal{P}^G$

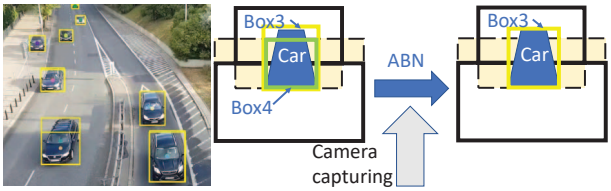
Input: $q(0) \leftarrow 0$, $\mathbf{X}_t$ is the list of N cameras.
Initialize $\mathbf{X}_0$: All inference in local cameras.
for slot $t = 0, 1, \ldots, T$ **do**
  repeat
    Randomly select a camera i, randomly select a region of a frame that needs to be detected, and offload it to a new camera j for processing.
    Then, update $\widehat{X_t^i}$ and $\widehat{\mathbf{X}_t}$.
    Solve $\mathcal{P}^L$ using $\widehat{\mathbf{X}_t}$ to obtain the objective function value $\widehat{g}$.
    Compute $\eta \leftarrow \frac{1}{1+e^{\left(\frac{\widehat{g}-g}{\tau}\right)}}$.
    $\mathbf{X}_t \leftarrow \widehat{\mathbf{X}_t}$ with probability η.
  until The iteration has reached the iteration rounds, or the objective function value has not significantly improved in the last 10 iterations ($|\widehat{g} - g| < 0.01$).
  Using the optimal scheduling policy $\mathbf{X}_t$
  $q(t+1) = \max\left[q(t) + (T_t - T_{\max}), 0\right]$

Algorithm 1 summarizes the solution: for each frame f in time slot t, a region of camera i is randomly selected and offloaded to a new camera j for processing, leading to $X_t^i[f, r] = j$. After obtaining new scheduling decision $\widehat{\mathbf{X}_t}$, we calculate its objective value $\widehat{g}$. Based on this, the transition probability η is computed, and a probabilistic decision is made to determine whether to retain the new scheduling decision as the temporary optimal solution. This process repeats until the termination conditions are met.

VI. ABN Module Design

After the Region Scheduler makes offloading decisions, cameras detect assigned regions and transmit the results. Within a single frame, detection results may come from multiple sources, including models from different cameras or optical flow-based tracking. As a result, objects may be enclosed by multiple bounding boxes, as shown in Fig. 7a, which can negatively impact detection precision. Therefore, it is crucial to refine the bounding box selection process to retain only the most complete and representative detections.

To address this, we introduce the ABN module, which effectively filters out redundant bounding boxes crossing regional boundaries. Unlike conventional Non-Maximum Suppression (NMS), ABN prioritizes bounding boxes based on area, retaining the largest one as it typically provides the most complete representation. This approach eliminates redundant and less informative boxes, improving efficiency and detection accuracy. As illustrated in Fig. 7b, when faced with overlapping bounding boxes, ABN retains Box3, which better encloses the object, while discarding Box4, which is redundant. By leveraging area as a selection criterion, ABN mitigates the negative impact of redundant bounding boxes, enhancing the robustness of region-based detection.



(a) Redundant boxes (b) The principle of ABN

Fig. 7: The process of ABN

VII. Implementation and Evaluation

A. Implementation

Our simulation evaluates the on-device performance of our method deployed on COTS hardware, including one edge server (NVIDIA RTX 3090 GPU, typical power consumption 350 W) and four edge devices (each with an NVIDIA Jetson Nano B01, typical power consumption 10–15 W). Network bandwidths are independently configured between devices in the range of 16–80 Mbps [20], reflecting edge-computing scenarios: lower bandwidth for the edge server (16 Mbps), and higher bandwidth for cameras (80 Mbps) to compensate for their limited compute power. The Region Scheduler runs on a controller (Python 3.8, NVIDIA MX230 GPU) at 130 ms per 1-second slot. YOLOv8l is deployed on the edge server, while YOLOv8m/s/n and YOLOv5nu run on CAM 1–4, all equipped with optical flow trackers.

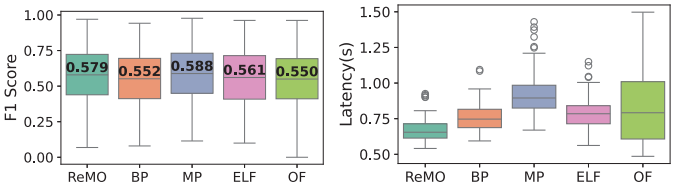
Following prior work, system parameters are precomputed and stored in a lookup table. Constants such as R_i, M_r, M_0, and α are fixed in advance, while $s_{i,M}$, s_0, a_i, and a_0 are estimated offline. The tracking drift factor $\phi_{i,t}$ is periodically updated during experiments by comparing detection and tracking results across frames. Detection threshold Q and padding ratio are adjusted online via lightweight heuristics.

B. Dataset and Baselines

Dataset: We evaluate our system on two datasets. The first includes four 720p, 25 FPS surveillance videos from YouTube [17], [21]–[23], covering diverse road conditions. We evaluate on synchronized 5-minute segments, using YOLOv8x detection results as the ground truth. The second is the AI City Challenge dataset [15], which captures real-world traffic scenes in a North American city. It contains 1955 synchronized frames per camera at 1080p and 10 FPS, with deployments spanning long roads and complex intersections.

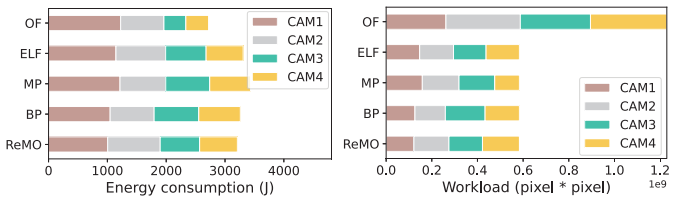
Baselines: We compare ReMO with the five baselines:

- **Bandwidth-Prior (BP):** Offloads excess workload to the camera with the highest available bandwidth.
- **Model-Prior (MP):** Assigns regions exhibiting maximal inter-frame flag variation (i.e., detection necessity) to devices hosting the best models.
- **Offload-Frame (OF):** Omits frame partitioning and directly applies the same scheduling algorithm to offload entire frames, detaching the Region Generator.
- **ELF:** [11] Predicts ROIs using a lightweight recurrent module and distributes detection tasks across edge servers based on previous latency. For fair comparison, we configure ELF's load-balancing mechanism for camera-to-camera offloading.
- **EHCI:** [14] A non-DBT approach that expands previous-frame bounding boxes with padding to form ROIs, which are offloaded to the edge server for throughput maximization under latency constraints. We evaluate accuracy and latency under identical conditions.



(a) Accuracy (b) Latency

Fig. 8: Comparison of accuracy and latency



(a) Energy consumption (b) Load balancing effect

Fig. 9: Comparison of energy consumption and load balancing

C. Baselines Comparison

Accuracy and Latency: As shown in Fig. 8, ReMO not only achieves superior accuracy (0.579), outperforming Bandwidth-Prior (0.552) and ELF (0.561), but also attains the lowest latency (0.66s) among all methods. The heuristic-based Bandwidth-Prior lacks adaptability to network and content dynamics, while ELF's backward-looking allocation strategy yields even poorer performance. Compared to Offload-Frame (0.799s), ReMO reduces latency by 17.4% with a 0.02 accuracy increase, validating that partitioning frames into regions effectively minimizes delays while improving accuracy. Model-Prior maximizes accuracy (0.588) by always selecting the highest-accuracy models, but incurs high latency (0.896s). In contrast, ReMO reduces this delay by 26.3% with only 1.5% accuracy degradation, achieving better balance between both metrics. These results confirm ReMO's ability to balance high accuracy with real-time performance through adaptive scheduling.

As illustrated in Fig. 10, under the same full-frame inference interval (e.g., 25 frames), ReMO achieves a 1.2% higher F1-score (0.579 vs. 0.572) and reduces latency by 26.0% (0.681s vs. 0.92s) compared to EHCI. This improvement stems from ReMO's region-based detection, which avoids object loss common in EHCI's method of expanding ROIs from previous frames. Moreover, while EHCI's centralized strategy introduces high transmission overhead for region aggregation, ReMO's peer-to-peer paradigm ensures full scene coverage with low-latency localized processing.

Energy Consumption: As shown in Fig. 9a, ReMO demonstrates competitive energy efficiency while maintaining accu-

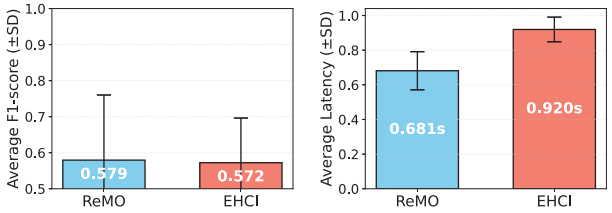


Fig. 10: Comparison with EHCI

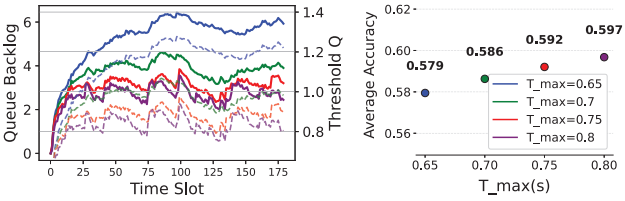


Fig. 11: Impacts of $T_{\max}$

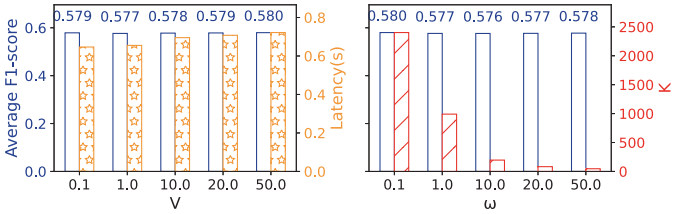


Fig. 12: Impacts of weight parameters

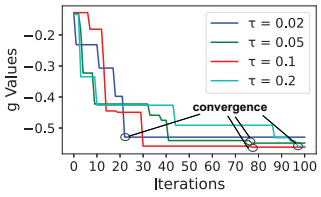


Fig. 13: g for different τ

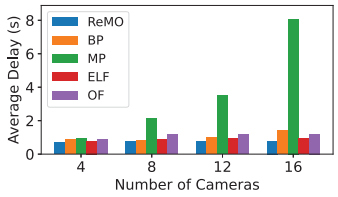


Fig. 14: N-Camera impact

racy. Although Offload-Frame achieves the lowest consumption by sacrificing accuracy through full-frame processing, ReMO reduces energy usage by 1.5% compared to Bandwidth-Prior , 6.5% versus Model-Prior, and 3.0% relative to ELF. This balanced performance stems from ReMO's dynamic workload distribution and intelligent device utilization. In contrast, Bandwidth-Prior and ELF exhibit suboptimal energy efficiency due to their limited adaptability, while Model-Prior's accuracy-first approach results in 6.9% higher consumption than ReMO. These results highlight ReMO's ability to optimize the accuracy-efficiency trade-off.

Load Balancing Effect: As shown in Fig. 9b, in Offload-Frame, since all devices perform full-frame detection, the total workload is the highest, though individual device loads remain relatively balanced. ReMO ensures fair workload distribution based on device capabilities to minimize system latency, but unlike ELF—which enforces an overly uniform (yet suboptimal) offloading. In contrast, Model-Prior and Bandwidth-Prior exhibit poor load balancing, as they overburden either the most computationally capable device (Model-Prior) or the one with the best bandwidth (Bandwidth-Prior). This uneven distribution increases overall latency, further validating ReMO's superiority in workload scheduling.

D. Exploring the Impact of Parameters

Delay Threshold T_{max}: As depicted in Fig. 11, it is observed that as the delay queue backlog increases (solid line), the detection threshold Q rises gradually (dashed line), reducing the number of regions requiring detection until both the queue backlog and Q eventually stabilize. The figure also reveals that increasing T_{max} yields three key effects: (1) the queuing delay stabilizes earlier, (2) the maximum queue length decreases, and (3) accuracy improves proportionally (gaining 0.012-0.016 per 0.1s T_{max} increment). These results confirm that properly configured T_{max} parameters can effectively balance latency increases and accuracy gains.

Parameters V and ω: As illustrated in Fig. 12, the parameter V controls the latency-utility trade-off. Increasing V (from 0.65 to 0.72) leads to higher latency but yields a more utility-optimized scheduling policy, while decreasing V favors low-latency operation. Accuracy remains largely unaffected by V, as it is primarily determined by the detection threshold Q, which decides whether a region is inferred. Meanwhile, the parameter ω governs the accuracy–load balancing trade-off: larger ω improves load balancing (reducing K by $\sim$30%) with only minor accuracy changes, again due to the dominant influence of Q. Hence, appropriate tuning of V and ω enables the system to adapt to varying application requirements.

Hyperparameter τ: Fig. 13 presents the convergence curve for iteratively searching for the optimal solution. When τ is relatively small (e.g., τ=0.02), convergence occurs within 23 iterations, reaching -0.53. Conversely, for a larger τ (e.g., τ = 0.2), convergence requires over 90 iterations but achieves a lower value of -0.57. Smaller τ speeds convergence; larger τ avoids suboptimal solutions. Here, a moderate value of $\tau = 0.1$ is chosen to strike a balance between exploration capability and accelerated convergence.

E. Analysis of Scalability

As shown in Fig. 14, camera scalability experiments (4/8/12/16 cameras) reveal ReMO's consistent latency advantage, with only a modest 0.1s increase (from 0.68s to 0.78s) as the number of cameras scales up. In contrast, other methods suffer more pronounced degradation: Model-Prior incurs a dramatic 4.07s increase (from 0.93s to 5.0s), Bandwidth-Prior rises by 0.55s (0.86s to 1.41s), while ELF and Offload-Frame see increases of 0.15s (0.79s to 0.94s) and 0.35s (0.86s to 1.21s), respectively. At 16 cameras, ReMO maintains the lowest latency (0.78s), achieving 6.25× better scalability than Model-Prior and 1.75× better than Bandwidth-Prior. These results demonstrate ReMO's superior ability to scale with minimal performance degradation, making it well-suited for large-scale deployments.

VIII. Related Work

Spatial Redundancy Within Frames: EVA [10] uploads bounding boxes, rather than entire frames, as the basic transmission elements for server processing. Elf [11] employs a recurrent region proposal prediction algorithm to identify object candidate regions and segment video frames for detection. MRLL [24] finds optimal ROIs without user input by

analyzing confidence scores from a chosen Machine Learning (ML) object detector. ROI-DVC [25], an ROI-based video coding network, integrates an ROI perception strategy into both the training and inference processes. While RegenHance [26] also relies on prediction, it specifically addresses the computational bottleneck by proposing a macroblock-based region importance predictor that uses a segmentation model. However, these studies generate the RoIs based on time-consuming predicting methods, making them impractical for deployment in resource-limited edge environments.

Adaptive Load Balancing: CrossVision [9] adaptively balances workloads among intelligent camera networks to reduce latency. Distream [13] adapts to workload dynamics, achieving low latency, high throughput, and scalability in real-time video analytics, while balancing workloads between intelligent cameras and edge clusters. WMA [27] achieves workload balancing through a two-stage resource allocation strategy. VideoJam [28] employs a set of load balancers to dynamically distribute incoming video traffic across replicas without centralized coordination, independently adapting to various system configurations and camera setups to handle load variations. However, these studies do not comprehensively consider changes in video content and fluctuations in network bandwidth when making scheduling decisions.

IX. Conclusion

This paper investigates fine-grained workload balancing through region-based offloading for collaborative video inference at the edge. We presents ReMO, an online scheduling framework for fine-grained region-based offloading in collaborative edge video inference. By leveraging static region partitioning, ReMO reduces transmission overhead and improves workload balance. Experiments show ReMO reduces latency by 9.3–26.3% over state-of-the-art methods.

It is crucial to acknowledge the work's limitations. Specifically, this work focuses on static cameras. Extending region partitioning to dynamic camera scenarios remains future work.

References

[1] Y. Chen, X. Yuan, J. Wang, R. Wu, X. Li, Q. Hou, and M.-M. Cheng, "Yolo-ms: rethinking multi-scale representation learning for real-time object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[2] F. Yuan, G. Wang, Q. Huang, and X. Li, "A newton interpolation network for smoke semantic segmentation," *Pattern Recognition*, vol. 159, p. 111119, 2025.

[3] S. Qiao, H. Xu, C. Cao, W. Gong, S. Chen, and J. Liu, "Prismprompt: Layering prompt-enhanced cloud-edge collaborative language model towards healthcare," *IEEE Network*, 2025.

[4] C. Cao, Y. Ding, M. Dai, W. Gong, and X. Zhao, "Real-time cross-domain gesture and user identification via cots wifi," *IEEE Transactions on Mobile Computing*, 2025.

[5] W. Wu and W. Gong, "Concurrent wifi backscatter communication using a single receiver in iot networks," *Computer Networks*, vol. 258, p. 111029, 2025.

[6] N. Chen, S. Zhang, J. Wu, H. Huang, and S. Lu, "Spliceosome: On-camera video thinning and tuning for timely and accurate analytics," *IEEE Transactions on Networking*, 2025.

[7] N. Chen, S. Zhang, S. Zhang, Y. Yan, Y. Chen, and S. Lu, "Resmap: Exploiting sparse residual feature map for accelerating cross-edge video analytics," in *In Proceedings of IEEE INFOCOM*, 2023, pp. 1–10.

[8] H. Guo, S. Yao, Z. Yang, Q. Zhou, and K. Nahrstedt, "Crossroi: cross-camera region of interest optimization for efficient real time video analytics at scale," in *In Proceedings of ACM MMSys*, 2021, pp. 186–199.

[9] L. Zhang, Z. Lu, L. Song, and J. Xu, "Crossvision: Real-time on-camera video analysis via common roi load balancing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 5027–5039, 2023.

[10] Z. Wang, X. He, Z. Zhang, Y. Zhang, Z. Cao, W. Cheng, W. Wang, and Y. Cui, "Edge-assisted real-time video analytics with spatial–temporal redundancy suppression," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 6324–6335, 2022.

[11] W. Zhang, Z. He, L. Liu, Z. Jia, Y. Liu, M. Gruteser, D. Raychaudhuri, and Y. Zhang, "Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading," in *In Proceedings of ACM MobiCom*, 2021, pp. 201–214.

[12] Y. Yan, S. Zhang, X. Wang, N. Chen, Y. Chen, Y. Liang, M. Xiao, and S. Lu, "Visflow: Adaptive content-aware video analytics on collaborative cameras," in *In Proceedings of IEEE INFOCOM*, 2024, pp. 2019–2028.

[13] X. Zeng, B. Fang, H. Shen, and M. Zhang, "Distream: scaling live video analytics with workload-adaptive distributed edge intelligence," in *In Proceedings of ACM SenSys*, 2020, pp. 409–421.

[14] X. Wang, M. Shen, and K. Yang, "On-edge high-throughput collaborative inference for real-time video analytics," *IEEE Internet of Things Journal*, vol. 11, no. 20, pp. 33 097–33 109, 2024.

[15] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang, "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," in *In Proceedings of IEEE CVPR*, 2019, pp. 8797–8806.

[16] K. Wang, K. Wang, and S. Shen, "Flownorm: A learning-based method for increasing convergence range of direct alignment," in *In Proceedings of IEEE ICRA*, 2020, pp. 2109–2115.

[17] A. Khandelwal, "M6 motorway traffic," Oct. 2013, [Online]. Available: https://www.youtube.com/watch?v=MNn9qKG2UFI&list=PLJKyZ_NuOhJQzif2-6-Kq9OiOj_UjJWvi&index=6.

[18] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *In Proceedings of IEEE INFOCOM*, 2020, pp. 257–266.

[19] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE transactions on information theory*, vol. 59, no. 10, pp. 6301–6327, 2013.

[20] N. Kan, Y. Jiang, C. Li, W. Dai, J. Zou, and H. Xiong, "Improving generalization for neural adaptive video streaming via meta reinforcement learning," in *In Proceedings of ACM MM*, 2022, pp. 3006–3016.

[21] A. Khandelwal, "K road traffic video for object detection and tracking - free download now!" Jul. 2018, [Online]. Available: https://www.youtube.com/watch?v=MNn9qKG2UFI&list=PLJKyZ_NuOhJQzif2-6-Kq9OiOj_UjJWvi&index=8.

[22] ——, "Relaxing highway traffic," Jul. 2017, [Online]. Available: https://www.youtube.com/watch?v=MNn9qKG2UFI&list=PLJKyZ_NuOhJQzif2-6-Kq9OiOj_UjJWvi&index=5.

[23] ——, "4k road traffic video for object detetction and tracking," Jun. 2021, [Online]. Available: https://www.youtube.com/watch?v=MNn9qKG2UFI&list=PLJKyZ_NuOhJQzif2-6-Kq9OiOj_UjJWvi&index=2.

[24] M. Qiu, L. Christopher, S. Y.-P. Chien, and Y. Chen, "Intelligent highway adaptive lane learning system in multiple rois of surveillance camera video," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[25] X. Wu, P. Wang, and X. Wang, "Roi-dvc: A region-of-interest based deep video coding framework," in *In Proceedings of IEEE ICIP*, 2024, pp. 1967–1972.

[26] W. Wang, L. Mi, S. Cen, H. Dai, Y. Li, X. Fu, and Y. Liu, "Region-based content enhancement for efficient video analytics at the edge," in *In Proceedings of USENIX NSDI*, 2025, pp. 613–633.

[27] H.-T. Chen, Y. Chiang, and H.-Y. Wei, "Edge computing resource management for cross-camera video analytics: Workload and model adaptation," *IEEE Access*, 2024.

[28] Y. Faye, F. Faticanti, S. Jain, and F. Bronzino, "Videojam: Self-balancing architecture for live video analytics," in *In Proceedings of IEEE/ACM SEC*, 2024, pp. 149–163.