2024 IEEE Intelligent Vehicles Symposium (IV)
June 2-5, 2024. Jeju Island, Korea

Enhancing AR/VR Performance via Optimized Edge-based Object Detection for Connected Autonomous Vehicles

Daniel Mawunyo Doe*, Dawei Chen†, Kyungtae Han†, Jiang Xie‡, and Zhu Han*
*Electrical and Computer Engineering, University of Houston, Houston, TX, USA
†InfoTech Labs, Toyota Motor North America R&D, Mountain View, CA, USA
‡ Electrical and Computer Engineering, The University of North Carolina at Charlotte, Charlotte, NC, USA

***Abstract*—The rapid integration of augmented reality (AR) and virtual reality (VR) technologies into contemporary automotive development has led to unprecedented opportunities and challenges. This work addresses the integration of edge computing and AR/VR applications within connected autonomous vehicles, focusing on the pivotal role of object detection. The edge-assisted object detection problem is formulated as a constrained optimization problem, aiming to minimize the adverse effects on the object detection process. To solve the problem, we introduce an innovative edge-assisted algorithm, transmitting live camera frames to an edge server for detailed processing. Only essential detection data is then relayed to AR/VR devices, marking a significant advancement over existing strategies. Notable outcomes include a reduction in latency (averaging between 37.06% and 44.76%), enhanced data throughput (ranging from 27.66% to 41.18%), improved freshness loss (between 36.36% and 69.57%), and a frame loss reduction to 7.5%, surpassing baseline methods by 6.5% to 36%. These findings underscore the potential of this methodology for optimizing AR/VR applications in vehicular environments.**

***Index Terms*—AR/VR integration, connected autonomous vehicles, object detection, edge computing, optimization.**

I. Introduction

The recent growth of augmented reality (AR) and virtual reality (VR) technologies integration in contemporary automotive development has been unprecedented. Cutting-edge solutions are seamlessly integrating AR features into vehicles, such as incorporating directions onto windshields for enhanced driver focus and information dissemination[1]. Real-time environmental data is now accessible to drivers, providing insights on traffic, pedestrians, and more. Simultaneously, rigorous training and testing of autonomous driving software are being conducted with the aid of AR/VR technologies[2]. These developments, together with immersive passenger experiences, are transforming the automotive industry in significant ways, elevating its economic and functional dimensions [1].

The centerpiece of this AR/VR evolution lies in object detection [1]. Its role is paramount in ensuring the safety, efficiency, and optimization of AR/VR applications within connected autonomous vehicles (CAVs). However, the integration of object detection is not without its challenges. Key among them are the constraints associated with bandwidth, the substantial energy demands, and the issues of latency arising when performing object detections [2]. As data gets processed and relayed, these issues amplify, potentially tarnishing the smooth operation of AR/VR systems in the vehicular context.

In the literature, several methodologies have been explored to address these challenges. For instance, in [3]–[5], the authors proposed onboard systems for AR/VR object detection in autonomous vehicles. However, these approaches encountered significant resource utilization and latency issues, particularly in dynamic environments [6]. In another project available on GitHub[3], cloud or edge servers were leveraged to tackle the computational demands of AR/VR integration. While these approaches improved onboard computational intensity, they faced challenges associated with constant frame exchanges between devices and servers, particularly in areas with limited network connectivity. Similarly, in the works by [7], [8], edge computing was employed to manage computational demands with a primary focus on power conservation. Although these works reported reduced device power consumption, they did not fully achieve real-time accuracy in object detection and also encountered frame exchange challenges between devices and servers. Furthermore, an important issue highlighted in the existing work, especially when using devices like the HoloLens, is the occurrence of frame drops[4]. The authors in [9], [10] recognized this challenge but did not provide any comprehensive solutions.

[1]https://m.youtube.com/watch?v=AXf3WoCAbxM
[2]https://nvidianews.nvidia.com/news/nvidia-introduces-drive-constellation-simulation-system-to-safely-drive-autonomous-vehicles-billions-of-miles-in-virtual-reality
[3]https://github.com/toolboc/IntelligentEdgeHOL, https://github.com/doughtmw/YoloDetectionHoloLens-Unity
[4]https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/quality-fundamentals

In this light, we propose an optimized methodology harnessing the potential of edge computing. Expressly, we formulate the edge-assisted object detection problem as a constrained optimization problem, meticulously designed to minimize various negative impacts on the object detection process. In our solution, instead of directly employing the object detection model on HoloLens, which has its set of frame drops and latency challenges, we propose streaming the live camera video frames to an edge server for object detection, which optimizes the object detection process. Next, we only transmit the essential object detection data (e.g., bounding boxes, class names, confidence) back to the HoloLens for rendering, which dramatically diminishes bandwidth usage and alleviates latency concerns. Moreover, the edge servers are equipped with potent GPUs to ensure improved accuracy in object detection, offer reduced power consumption for AR/VR devices, and enhance user data security, marking our solution a promising step forward in the seamless integration of AR/VR into autonomous vehicles. To our knowledge, this work is one of the first attempts to leverage edge computing for optimized object detection in AR/VR-integrated CAVs. The main contributions of this work are as follows:

- We formulate the edge-assisted object detection problem as a constrained optimization problem, aiming to minimize the adverse effects on the object detection process.
- We introduce a novel framework that streams live camera frames to edge servers for object detection and transmits only essential data to the HoloLens, which conserves bandwidth, ensures freshness, and mitigates latency and frame drop challenges.
- We validate the efficiency and effectiveness of our methodology through comprehensive experiments, establishing it as a pioneering advancement for integrating AR/VR in autonomous vehicles.

The remainder of this paper is organized as follows: Section II delves into the description of the AR/VR system model in the context of autonomous vehicles. Our novel edge-assisted approach for optimized object detection in AR/VR systems is elucidated in Section III. An evaluation of the system's performance through various experiments and simulations is presented in Section IV. The manuscript is concluded in Section V, encapsulating the salient points of our discourse.

II. System Model

A. System Architecture

Our system architecture consists of three integral components: AR/VR headsets (Microsoft HoloLens), edge servers, and the CAVs themselves. The AR/VR headsets function as the user interface, capturing live video frames from the CAV's camera and rendering the object detection data returned by the edge servers. The edge servers are strategically located near the CAVs' operational regions to provide high-performance GPUs and CPUs, to handle the intensive computational task of object detection on the video frames. We incorporate a state-of-the-art vehicle-to-vehicle (V2V) multi-hop relaying framework and vehicle-to-infrastructure (V2I), akin to the approaches seen in [8]. These V2V and V2I architectures ensure efficient data routing and offloading decisions between CAVs, edge, and cloud servers.

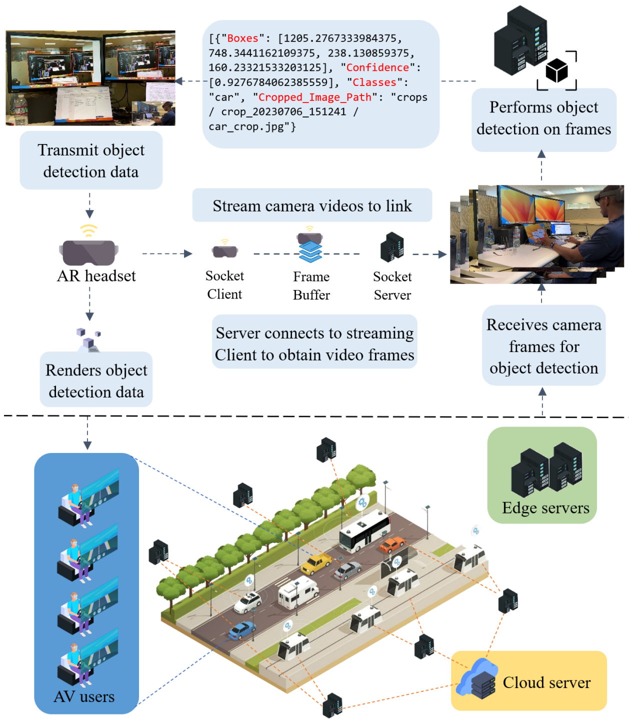

```
[{"Boxes": [1205.2767333984375,
748.3441162109375, 238.130859375,
160.23321533203125], "Confidence":
[0.9276784062385559], "Classes":
"car", "Cropped_Image_Path": "crops
/ crop_20230706_151241 /
car_crop.jpg"}
```
Performs object detection on frames
Transmit object detection data
Stream camera videos to link
AR headset
Socket Client
Frame Buffer
Socket Server
Server connects to streaming Client to obtain video frames
Receives camera frames for object detection
Renders object detection data
Edge servers
AV users
Cloud server

Figure 1: Our proposed edge-assisted object detection enhancement system architecture.

In our proposed design, we address the pivotal challenges of bandwidth, latency, frame loss, and data freshness while aiming to deliver superior object detection accuracy. The AR/VR headsets are optimized to concentrate solely on capturing and rendering data, conserving their computational resources. For object detection, we harness the power of the latest algorithm, YOLOv8, renowned for its accuracy and efficiency [11]. Given the superior computational capabilities of the edge servers in comparison to AR/VR devices, employing YOLOv8 allows for precise and timely object recognition. The live video frames are relayed from the AR/VR headsets to these servers. Following object detection, only crucial information like bounding boxes, confidence scores, and class labels are transmitted back to the headsets for rendering. Fig. 1 illustrates our proposed system architecture.

B. Network Model

We consider a system of H HoloLens integrated CAVs, E edge servers, and M cloud servers. Specifically, the sets of CAVs, edge servers, and cloud servers are defined as $\mathcal{H} = \{1, 2, \ldots, H\}$, $\mathcal{E} = \{1, 2, \ldots, E\}$, and $\mathcal{M} = \{1, 2, \ldots, M\}$, respectively. Each CAV captures frames F via its HoloLens camera sensors, which contain objects located in the environment. Let CAV h and edge server e be within a designated region K. CAV h possesses a computing power C_h, utilized for decoding results and rendering after edge processing. In contrast, edge server e is endowed with computing power C_e to perform object detection and results encoding on frames F_k.

Frames F_k from the HoloLens sensors in the k-th sub-region can be defined as:

$$F_k = \begin{cases} F_{h,k} = x_{h,k} \cdot \sigma_{h,k} \cdot \frac{F}{K}, \\ F_{e,k} = x_{e,k} \cdot \sigma_{e,k} \cdot \frac{F}{K}, \end{cases} \quad (1)$$

where $F_{h,k}$ and $F_{e,k}$ represent the frame data from CAV h and edge server e, respectively. $x_{h,k}$ and $x_{e,k}$ identify the CAVs or edge servers closest to the sub-region k for data processing. $\sigma_{h,k}$ and $\sigma_{e,k}$ are the sensing variables for CAV h and edge server e, respectively.

The computation time t_k in sub-region k taken by CAV h or edge server e can be denoted by:

$$t_k = \begin{cases} t_{h,k} = \frac{F_{h,k} \cdot C_{F_{h,k}}}{C_h}, \\ t_{e,k} = \frac{F_{e,k} \cdot C_{F_{e,k}}}{C_e}. \end{cases} \quad (2)$$

The wireless transmission rate from CAV h to edge server e separated by distance $d_{h,e}$ is expressed as:

$$R_{h,e} = b_{h,e} \cdot \log\left(1 + \frac{P_{h,e} \cdot |Q_{h,e}|^2 \cdot (d_{h,e})^{-\varphi_{h,e}}}{N_{h,e}}\right), \quad (3)$$

where $b_{h,e}$, $P_{h,e}$, $Q_{h,e}$, $\varphi_{h,e}$, and $N_{h,e}$ denote the allocated bandwidth, transmission power, channel fading coefficient, path-loss exponent, and noise power between the CAV and the edge server, respectively. For transmissions from an edge server e to a cloud server m, separated by distance $d_{e,m}$, the rate is given by:

$$R_{e,m} = b_{e,m} \cdot \log\left(1 + \frac{P_{e,m} \cdot G_{e,m}}{I + N_{e,m}}\right), \quad (4)$$

where $b_{e,m}$, $P_{e,m}$, $G_{e,m}$, and $N_{e,m}$ represent vehicle h's allocated bandwidth, transmission power, gain over transport and core networks influenced by direct radio communications, and noise power, respectively.

Suppose that an CAV begins frame offloading to the edge, the total transmission data size (e.g., query, response, or status update) is denoted as Π_{F_k} and can be expressed as:

$$\Pi_{F_k} = \begin{cases} \Pi_{F_{h,k}} = F_{h,k} \cdot \left(|\frac{L_h}{\lambda}| + 1\right), \\ \Pi_{F_{e,k}} = F_{e,k} \cdot \left(|\frac{L_e}{\lambda}| + 1\right), \end{cases} \quad (5)$$

where $\Pi_{F_{h,k}}$ and $\Pi_{F_{e,k}}$ denote the transmission data from h and e, respectively. $|\frac{L_h}{\lambda}|$ and $|\frac{L_e}{\lambda}|$ represent the number of relay hops between h and e. Relay hops ($\frac{L}{\lambda}$) are considered for V2V communications only, with L denoting the total relay distance and λ the effective range of a single hop. These hops facilitate data transmission through multiple relay nodes, with the exact count influenced by network topology and configuration. Lastly, the round trip time (RTT) t_{RTT} during an update cycle T is expressed as:

$$t_{RTT} = \begin{cases} \sum_{k=1}^{K} (t_{F_{h,k}} + t_{h,k} + t_{h,e} + t_{e,m}), \\ \sum_{k=1}^{K} (t_{F_{e,k}} + t_{e,k} + t_{h,e} + t_{e,m}), \end{cases} \quad (6)$$

which accounts for the time taken to process frame data on h, transmit from CAV h to edge server e, and relay from edge server e to cloud server m.

C. Utility Model

Our framework, as introduced in Section II-A, leverages the edge-cloud synergy to enhance AR/VR experiences via HoloLens-integrated CAVs. This enhancement focuses on optimizing the object detection process through efficient data transmission and rendering. To quantify this enhancement, we introduce the following definition.

Definition 1 (*Object Detection Enhancement (ODE)*). *ODE captures the efficiency gains achieved through optimizing object detection, taking into account the factors of frame transmission size, frame freshness, transmission times, and potential data loss during communication.*

The ODE is influenced by:

1) **Size of Transmission Frame**: Represented as F_{size_k} for frame F_k.
2) **Freshness of Frame**: Crucial for real-time AR/VR experiences. The freshness loss θ_{F_k} for frame F_k is given by:

$$\theta_{F_k} = \frac{t_{RTT}}{\delta t_{F_k}}, \quad (7)$$

where δt_{F_k} denotes the last update interval of the frame, $\delta t_{F_k} = \frac{d_{h,e}}{a}$ denotes the interval for data updates and a is the moving speed of the vehicle.
3) **Transmission Time**: Similar to t_{RTT} in (6).
4) **Frame Loss**: Losses can occur at the HoloLens device, during HoloLens-to-edge transmission, and

in the edge-to-HoloLens return transmission. This frame loss L_{F_k} is expressed as:

$$L_{F_k} = \begin{cases} L_{F_{h,k}}, & \text{if originating from HoloLens,} \\ L_{F_{e,k}}, & \text{if during HoloLens to edge,} \\ L_{F_{r,k}}, & \text{if during edge to HoloLens return.} \end{cases} \tag{8}$$

Our object detection utility O_{F_k} for frame F_k is formulated as inversely proportional to the aggregated negative effects, such as freshness loss, transmission time, data loss, and transmission size. Specifically:

$$O_{F_k} = \frac{1}{\alpha_1 \theta_{F_k} + \alpha_2 t_{RTT} + \alpha_3 L_{F_k} + \alpha_4 F_{size_k}}, \tag{9}$$

where $\alpha_1, \alpha_2, \alpha_3,$ and α_4 are weights reflecting the importance of each factor in the model. These weights can be adjusted to focus on specific objectives, such as minimizing latency or data loss, and can be calibrated through empirical evaluation, regression analysis, or sensitivity analysis. By integrating these elements, we offer a comprehensive metric to gauge the efficiency of object detection, directly influencing the quality of AR/VR experiences.

III. Edge-assisted Object Detection Enhancement Algorithm (EODEA) Formulation

The rapid advancement in HoloLens technology demands real-time object detection enhancement, especially for applications involving AR/VR. Given this, we introduce an edge-assisted strategy tailored to this requirement. Precisely, our ODE problem is formulated after the following definition.

Definition 2 (*Edge-assisted Object Detection Enhancement (EODE)*). *Given a HoloLens device and edge servers, how can we enhance the object detection process, ensuring minimized frame loss, optimized data transmission time and size, while maintaining result freshness for user satisfaction?*

The EODE aims to minimize the composite measure of various negative impacts on the detection process under constraints like freshness loss, frame loss during transmission, data transmission time, and size. The EODE optimization problem can be represented as:

$$\min_{\theta_{F_k}, t_{RTT}, L_{F_k}, F_{size_k}} \quad O_{F_k} \tag{10a}$$

$$\text{s.t.} \quad \theta_{F_k} \leq \bar{\theta}_{F_k}, \tag{10b}$$

$$t_k < t_{RTT} \leq \bar{t}_{RTT}, \tag{10c}$$

$$L_{F_k} \leq \bar{L}_{F_k}, \tag{10d}$$

$$F_{size_k} \leq \bar{F}_{size_k}, \tag{10e}$$

$$F_{h,k} \leq F_h, \quad F_{e,k} \leq F_e, \tag{10f}$$

$$C_{F_{h,k}} \leq C_h, \quad C_{F_{e,k}} \leq C_e, \tag{10g}$$

$$R_{h,e} \leq \bar{R}_{h,e}, R_{e,m} \leq \bar{R}_{e,m}, \tag{10h}$$

where $\bar{\theta}_{F_k}, \bar{L}_{F_k}, \bar{F}_{size_k}, \bar{R}_{h,e}, \bar{R}_{e,m}$, and $\bar{t}_{RTT}$ represent the maximum allowed values for freshness loss, frame loss, size of transmission frame, data transfer rates $R_{h,e}$ and $R_{e,m}$, and transmission delay, respectively. Essential to this optimization are several constraints. The freshness loss of a frame, θ_{F_k}, constrained by (10b), should be within a predefined threshold, $\bar{\theta}_{F_k}$. Additionally, transmission time t_{RTT} has defined boundaries given by (10c). The potential loss of frames, constrained by (10d), should not exceed a specific threshold, $\bar{L}_{F_k}$, while the size of the transmitted frame, constrained by (10e), is expected to remain below $\bar{F}_{size_k}$. Furthermore, there are technological and infrastructural constraints associated with the number of frames, capacities of both the HoloLens device and edge servers, as well as the transmission rates, encapsulated in (10f), (10g), and (10h), respectively.

Due to the intricate nature of our optimization problem and the multiple constraints and parameters it involves, the problem is computationally challenging, likely falling into the NP-hard category. For instance, meeting the conditions set forth in (10b)–(10h) demands a polynomial time complexity, thereby classifying the EODE problem as NP-hard. This classification is corroborated by drawing on insights from Theorem 1 in [12], where the problem's complexity is likened to that of the multiple-choice knapsack problem via a reduction technique. This particular reduction is itself NP-complete and generally leads to solutions that are computationally intensive. As a result, finding the optimal solution within a practical timeframe can be quite challenging. Hence, approximation algorithms or heuristic methods often serve as more feasible approaches for obtaining satisfactory solutions.

To overcome the optimization challenge, we introduce a heuristic algorithm in Alg. 1 designed to minimize the objective function while respecting the various constraints. For tackling the issues related to frame loss, a server-client relationship is established between the HoloLens device and the edge server. Here, the server consistently acquires frames, whereas the client first gathers them into a buffer before forwarding. Any losses are managed on the client side. For optimizing data transmission and its size, we transmit only the essential object detection data to the HoloLens. This eliminates the need for superfluous encoding and decoding. As a solution for the freshness loss, we put forth the freshness preservation scheme (FPS). Under this scheme, if the time taken for processing and transmitting a frame exceeds a predetermined threshold, or if a sudden movement in the user's head is detected (through built-in sensors or eye tracking), the frame's outcome is discarded. This ensures that users are presented only with pertinent information.

Algorithm 1: Edge-assisted Object Detection Enhancement Algorithm (EODEA)

```
Input : $F_{h,k}$ (Frames from HoloLens), edge
        server parameters $C_{F_{e,k}}$, $\bar{\theta}_{F_k}$ (freshness
        threshold), $\bar{L}_{F_k}$ (max allowed frame
        loss), $\bar{F}_{size_k}$ (max allowed frame size)
Output: $O_{F_k}$ (Enhanced object detection results)
1  Initialization:
2  Establish asynchronous server-client connection
     between $e$ (edge server) and $h$ (HoloLens)
     ensuring resource optimization;
3  Server continuously receives frames $F_{h,k}$;
4  Client buffers and forwards frames, handling
     losses $L_{F_k}$;
5  for each frame $F_{h,k}$ do
6      Deserialize object detection results from
         JSON format;
7      Extract 2D frame coordinates and transform
         into 3D world coordinates;
8      Optimize data transmission by sending only
         object detection data serialized as JSON;
9      if transmission time $t_{RTT} > \bar{\theta}_{F_k}$ or sudden
         user movement detected using built-in
         sensors then
10         Discard frame result and log as $L_{F_k}$;
11     end
12 end
13 Return $O_{F_k}$ (enhanced object detection results);
```

Our proposed EODEA exhibits a time complexity of $\mathcal{O}(N)$ and a space complexity of $\mathcal{O}(N)$, where N represents the number of frames processed. In EODEA, initialization and returning results contribute constant time complexity $\mathcal{O}(1)$, while the main computational load resides in the for loop iterating through frames. Within this loop, operations like deserialization, data transformation, and data transmission are linear with the number of frames, thus yielding the $\mathcal{O}(N)$ complexity. These complexities show the algorithm's resource requirements and scalability concerning the number of frames processed.

IV. Experiment Results and Analysis

A. System Configuration

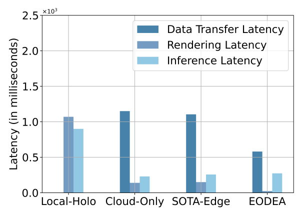
Our experimental setup leverages a hybrid system that combines the computational prowess of edge servers with the immersive capabilities of AR devices.

Edge Servers: Our framework is built on NVIDIA Jetson Xavier platforms in Python environments, chosen for their balance of power and efficiency. The computational core of our system is rooted in Python environments on NVIDIA Jetson Xavier platforms with Ubuntu 18.04 operating system. Each of these systems is equipped with 32GB of RAM and 512GB of SSD storage. While our selection of this specific hardware demonstrates the system's capabilities under certain conditions, we acknowledge that variations in hardware can influence performance outcomes.

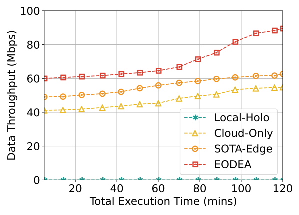
AR Devices: For the AR interface, we employ the Microsoft HoloLens 2 programmed using Unity 2020.3.48f1 and C#. This choice reflects a balance between current technological capabilities and availability. We recognize that alternative AR devices may yield different results, and our future research aims to assess the system's performance across a broader range of AR technologies.

Parameters and Variables: To ensure reproducibility, we simulate with $H = 50$ augmented vehicles (CAVs) each equipped with HoloLens, and these CAVs are connected to $E = 10$ edge servers and $M = 5$ cloud servers. Frames are captured at a rate of 30 fps, CAVs and edge servers are characterized by their computing power, with $C_h = 8$ TFLOPs for the CAVs and $C_e = 50$ TFLOPs for edge servers. We set $R_{h,e}$ and $R_{e,m}$ at 1 Gbps with average distance ($d_{h,e}$) and ($d_{e,m}$) set to 100m and 10km. For the integrity and freshness of data, θ_{F_k} is allowed to vary between 0.4 and 0.9. Meanwhile, L_{F_k} is set at 5%, indicating the frame loss rate. The coefficients $\alpha_1, \alpha_2, \alpha_3$ and α_4 are determined to be $0.1, 0.15, 0.2,$ and 0.25, respectively, to demonstrate variance in sensitivity. We set $\lambda = 100m$, $L_h = 300m$, and $L_e = 500m$ to reflect urban V2V and V2I communication scenarios.

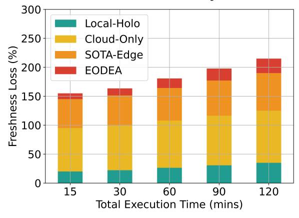
Comparative Analysis and Generalizability: This configuration is designed to benchmark our system against existing solutions, offering a unique combination of edge computing and AR technology. We aim to provide a detailed comparison in future work, highlighting the strengths and limitations of our approach. The specific assumptions and parameters used in our experiments are a starting point, and we acknowledge the need for further research to understand how these factors impact the generalizability of our results.



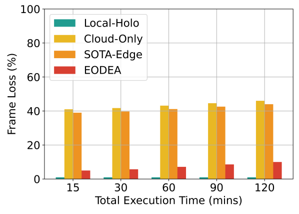
(a) Various Latency (ms)



(b) Data Throughput (Mbps)



(c) Freshness Loss (%)



(d) Frame Loss (%)

Figure 2: Performance comparison of our proposed EODEA and baseline schemes

B. Implementation Discussions

1) Latency Analysis: In Fig. 2a, a thorough latency analysis (e.g., the time delay experienced from sending a processing request (like a frame) to receiving its respective response.) reveals significant insights into the comparison of Local-Holo[5], Cloud-Only[6], SOTA-Edge[7], and the proposed EODEA. The Cloud-Only scheme incurs the highest data transfer latency, peaking at 2 seconds, attributed to the long-distance transmissions necessary for cloud computing. Conversely, Local-Holo exhibits minimal data transfer latency due to the absence of data transmission, but it struggles with high rendering and inference latencies. This surge in latency is due to the limited computational resources available on the HoloLens device, making it less adept at handling heavy computational tasks efficiently. Examining the data, EODEA consistently demonstrates superior performance, maintaining a low data transfer latency that does not exceed 1.01 seconds at the 120-minute mark. This efficiency arises from the transmission of serialized JSON data, which minimizes the data load and enhances speed. Rendering latency for EODEA remains impressively low (23.4ms) across all timestamps, a reflection of the optimized approach in handling graphical data. In the realm of inference latency, EODEA showcases modest and stable values, benefiting from streamlined data management and efficient processing mechanisms in place.

2) Throughput Analysis: In Fig. 2b, we elucidate the data throughput (e.g., the volume of data dispatched from the AR device to the edge servers and subsequently to the cloud servers within a given duration.) capabilities across the four distinct setups: Local-Holo, Cloud-Only, SOTA-Edge, and EODEA. From Fig. 2b, the Local-Holo setup presents an invariant throughput at zero, which is anticipated since no data transmission takes place beyond the device itself. On the contrary, Cloud-Only tends to lag behind the others, signifying a moderate throughput level. This diminished throughput for the Cloud-Only scheme can be primarily ascribed to the bottlenecks introduced by long-distance data transmissions inherent to cloud-centric models. From Fig. 2b, the SOTA-Edge setup exhibits an enhanced throughput performance in relation to Cloud-Only. This can be credited to its closer proximity to data sources, facilitating a more robust data flow. However, the crowning jewel of the analysis remains the EODEA approach. The EODEA curve consistently surpasses the rest, reaching peak values of approximately 89 Mbps by the 120-minute mark. The framework's design is explicitly optimized to handle voluminous data flows, ensuring the highest throughput among all transmitting schemes.

3) Freshness Loss Analysis: In Fig. 2c, the freshness loss (e.g., the proportion of frames that are lost amidst transmission or during processing.) is graphically represented for four different setups over time, ranging from 15 to 120 minutes. From Fig. 2c, it is evident that the Cloud-Only setup suffers the most significant freshness loss, reaching values near 90%. This high percentage is attributed to the substantial latency induced by remote computations. This latency leads to data being outdated by the time it is received, resulting in a stark deterioration of data freshness. On the other hand, the SOTA-Edge and Local-Holo setups showcase moderated freshness loss levels. Despite the SOTA-Edge's close proximity to data sources, which helps in reducing latency and thereby preserving some level of data freshness, it is not entirely exempt from freshness loss, peaking around 65%. This is attributed to the fact that processing may not be fully optimized in this setup. Local-Holo, while avoiding external data transmission, experiences a freshness loss due to its limited computational capabilities, reaching up to 35% loss, as inferred from the simulation. The EODEA setup emerges as the most efficient, maintaining the highest levels of data freshness by minimizing delay effectively. By optimizing data flows and computations, EODEA ensures rapid delivery of results, culminating in a minimal freshness loss, peaking at a mere 25%. This underscores the EODEA setup's capability in delivering timely and relevant data, making it highly suitable for applications where data freshness is paramount.

4) Frame Loss Analysis: Fig. 2d graphically represents frame loss (e.g., the decline in the relevancy of

[5]https://github.com/XRLabPomerania/Hololens-2-Object-Detection
[6]https://github.com/toolboc/IntelligentEdgeHOL
[7]https://github.com/doughtmw/YoloDetectionHoloLens-Unity

data over time.) over different intervals for four diverse setups. In this analysis, Cloud-Only setup manifests significant frame loss, escalating up to 46%. The SOTA-Edge setup demonstrates a tangible improvement, limiting frame loss to around 44%. This trend in results for Cloud-Only and SOTA-Edge setup arises due to the bottleneck created by the existing implementation from HoloLens application programming interface (API) for transmitting frames and extended-distance transmissions to a central cloud or edge, exacerbating the frame loss. This trend emphasizes the inefficacy of the Cloud-Only and SOTA-Edge setup in maintaining frame integrity, particularly in applications where real-time or reliable data transmission is non-negotiable. Local-Holo, on the other hand, is not applicable (N/A) for this metric as all its computations are local, eliminating the external data transmission, and thereby negating frame loss. EODEA setup stands out distinctly by ensuring the lowest frame loss, varying from 5% to 10%. This efficiency is owing to the system's optimization for large data flows and camera frame streaming service, ensuring the highest throughput among all the evaluated setups. This minimal frame loss highlights EODEA's robustness in maintaining data integrity during transmissions, underlining its suitability for applications demanding high reliability and real-time data communication.

V. Conclusion

In this study, we delved into the challenges of integrating AR/VR applications within autonomous vehicles, with a particular emphasis on the crucial task of object detection. we first formulated the edge-assisted object detection problem as a constrained optimization problem, targeting the minimization of adverse impacts on the object detection process. Addressing the challenges of solving our optimization problem, we introduced an innovative edge-assisted algorithm that directs live camera video frames to an edge server for in-depth processing. Subsequently, only vital detection data is sent back to devices like the HoloLens, signifying a substantial enhancement over previous methodologies. We observed a remarkable reduction in latency, with averages spanning from 37.06% to 44.76%. The system also demonstrated improved data throughput, fluctuating between 27.66% to 41.18%. There was a notable improvement in freshness loss, averaging between 36.36% and 69.57%, and frame loss was considerably reduced to 7.5%, outperforming baseline schemes by 6.5% to 36%. These results confirm the efficacy of our approach, paving the way for robust AR/VR implementations in vehicular settings.

References

[1] P. Zhou, J. Zhu, Y. Wang, Y. Lu, Z. Wei, H. Shi, Y. Ding, Y. Gao, Q. Huang, Y. Shi *et al.*, "Vetaverse: Technologies, applications, and visions toward the intersection of metaverse, vehicles, and transportation systems," *arXiv preprint arXiv:2210.15109*, Oct. 2022.

[2] M. Eswaran and M. R. Bahubalendruni, "Challenges and opportunities on ar/vr technologies for manufacturing systems in the context of industry 4.0: A state of the art review," *Journal of Manufacturing Systems*, vol. 65, pp. 260–278, Oct. 2022.

[3] O. El Marai, T. Taleb, and J. Song, "AR-based remote command and control service: Self-driving vehicles use case," *IEEE Network*, vol. 37, no. 3, pp. 170–177, Oct. 2023.

[4] D. Jallepalli, N. C. Ravikumar, P. V. Badarinath, S. Uchil, and M. A. Suresh, "Federated learning for object detection in autonomous vehicles," in *IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)*, Aug. 2021, pp. 107–114.

[5] C. Hu, R. Lu, and D. Wang, "FEVA: A federated video analytics architecture for networked smart cameras," *IEEE Network*, vol. 35, no. 6, pp. 163–170, Nov. 2021.

[6] H. Wang, Z. Wang, D. Chen, Q. Liu, H. Ke, and K. K. Han, "Metamobility: Connecting future mobility with the metaverse," *IEEE Vehicular Technology Magazine*, vol. 18, no. 3, pp. 69–79, Sep. 2023.

[7] H. Wang and J. Xie, "User preference based energy-aware mobile ar system with edge computing," in *IEEE INFOCOM conference on computer communications*, Jul. 2020, pp. 1379–1388.

[8] S. Liang, H. Wu, L. Zhen, Q. Hua, S. Garg, G. Kaddoum, M. M. Hassan, and K. Yu, "Edge yolo: Real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25 345–25 360, Mar. 2022.

[9] A. Ibrahim, J. B. Lanier, B. Huynh, J. O'Donovan, and T. Höllerer, "Real-time object recognition on the microsoft hololens," Technical report, UC Santa Barbara, Tech. Rep., 2017.

[10] R. Hammady, M. Ma, and C. Strathearn, "User experience design for mixed reality: a case study of hololens in museum," *International Journal of Technology Marketing*, vol. 13, no. 3-4, pp. 354–375, Jan. 2019.

[11] J. Terven and D. Cordova-Esparza, "A comprehensive review of yolo: From yolov1 to yolov8 and beyond," *arXiv preprint arXiv:2304.00501*, Apr. 2023.

[12] G. Papaioannou and I. Koutsopoulos, "Tile-based caching optimization for 360 videos," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '19, Jul. 2019, pp. 171–180.