DeVA: An Edge-assisted Video Analytics Framework for Depth Estimation

Shutong Chen*, Jingwen Yin, Ruichao Zhong, Fangming Liu, *Senior Member, IEEE*

***Abstract*—Edge-assisted video analytics frameworks, which offload vision-based tasks to edge servers, offer a promising approach to enhance accuracy while minimizing network resource overhead. However, these frameworks often overlook depth estimation, a critical task for applications like augmented reality and intelligent surveillance. Depth estimation, which calculates the distance between objects and the camera, generates depth images with unique characteristics, making existing approaches impractical or inefficient for video analytics in this context.**

In this work, we present DeVA, an edge-assisted video analytics framework for depth estimation that ensures accuracy with minimal network resource overhead. We examine the impact of various video analytics configurations, including resolution and quantization parameter (QP), on accuracy. Additionally, we analyze the region of interest (RoI) for depth estimation and propose methods for tracking RoI areas locally on the device. DeVA features an adaptive video encoding mechanism that dynamically adjusts the resolution for offloaded video and optimizes QPs for RoI and non-RoI areas. We implement DeVA and evaluate its performance using public video datasets. The results show that DeVA reduces 57.12% of the bandwidth overhead while keeping depth estimation errors within acceptable limits, demonstrating a great balance between accuracy and network resource usage.

***Index Terms*—edge-assisted video analytics, depth estimation, region of interest, video encoding**

I. Introduction

RECENT years have witnessed the rise of video analytics, which forms the foundation of augmented reality (AR), intelligent surveillance, and other advanced applications. Due to the substantial computational resources required for video analytics, it is common to offload these tasks to an edge server for the accuracy guarantee [1], [2]. However, existing approaches primarily focus on tasks like object detection and pose estimation, which provide discrete measures by identifying objects or locating keypoints of target objects. These methods often overlook another important video analytics task—depth estimation. Depth estimation provides a continuous measure (i.e., depth value) for each pixel, which is crucial for understanding the spatial arrangement of a scene. For example, AR applications rely on accurate depth estimation to enable realistic interactions between virtual objects and the real world [3]. Similarly, intelligent surveillance leverages depth information for improving the accuracy of human detection and anomaly detection [4], [5].

Existing edge-based video analytics frameworks achieve high accuracy of video analytics with low bandwidth overhead by strategically offloading the task from the client to the edge. For example, some approaches compress the resolution or filter offloaded frames to reduce the size of offloaded video [6]–[8]. To further optimize video compression, researchers have proposed fine-grained video encoding mechanisms that capture the region of interest (RoI) for the video analytics task, and apply high quality to the RoI and low quality to non-RoI areas [9]–[12].

However, most existing frameworks focus on "discrete" video analytics, such as object detection, and cannot be directly extended for depth estimation tasks. As detailed in Section II, object detection outputs a few bounding boxes with labels and scores, while depth estimation produces a depth image where each pixel encodes scene depth. For example, detecting a car requires only its location and label, whereas depth estimation must recover the precise depth of every pixel on its surface. The RoI areas[1] which have a significant effect on the depth estimation accuracy, cannot be fully captured by the detection model, potentially increasing the error of depth estimation, measured by the root mean squared error (RMSE), by 43.49%. Moreover, coarse-grained configurations, such as lower resolution and reduced quantization parameters (QPs)[2], have different degrees of impact on depth estimation and object detection. Based on our observations in Section II-B, the error of depth estimation can increase by more than 2.47 times when using coarse-grained configurations, while the intersection over union (IoU) for object detection reduces by 9%. As a result, it is impractical and inefficient to utilize a detection-RoI encoding framework for handling depth estimation tasks.

In this work, we propose DeVA, an edge-assisted video analytics framework for depth estimation designed to balance accuracy and bandwidth overhead. DeVA analyzes the RoI areas for depth estimation, dynamically adjusts the configurations for video analytics, including the resolution of encoded video and the QPs for RoI and non-RoI areas, and finally offloads the encoded video to the edge server for analysis.

This work was supported in part by the High-level Talent Program of Guangxi University under Grant A3070051017, the National Key Research & Development (R&D) Plan under grant 2022YFB4501703, the Major Key Project of PCL under Grant PCL2024A06 and PCL2022A05, and the Shenzhen Science and Technology Program under Grant RCJC20231211085918010. (Corresponding author: Shutong Chen)

S. Chen, J. Yin, and R. Zhong are with Guangxi University, China. E-mail: shutongcs@gxu.edu.cn, yjingwen1107@gmail.com, zhongshetin@foxmail.com.

F. Liu is with Peng Cheng Laboratory, and Huazhong University of Science and Technology, China. E-mail: fangminghk@gmail.com.

[1]Unless otherwise specified, the term RoI refers to RoI for depth estimation in the following discussion.

[2]QPs in video encoding control the degree of compression by adjusting the level of spatial detail retained in the encoded output [13]. A higher QP results in stronger compression and smaller file size but with reduced visual quality and potential degradation of analytic accuracy. Conversely, a lower QP preserves more detail, yielding better quality at the cost of a larger file size.

DeVA addresses the following three key challenges in reducing the error of depth estimation for the edge-assisted video analytics with minimal overhead:

The relationship between configuration and error of depth estimation is unclear. Although there has been some progress in video analytics, existing work does not thoroughly explore the factors affecting the error of depth estimation. Balancing the tradeoff between accuracy and bandwidth overhead is challenging without understanding these relationships. To address this challenge, we measure the impact of commonly used configurations, including resolution and QPs, on RMSE of depth estimation. Our measurement results reveal that reducing resolution significantly increases the error compared to changes in QPs. Additionally, the scenario (indoor or outdoor), the period (day or night), and the client's motion status (moving or static) greatly influence the configuration-RMSE relationship. DeVA introduces a lightweight content analysis approach. To be specific, it utilizes the relationship between pixel value in grayscale frame and the brightness to estimate the period. Moreover, it trains a ResNet18-based model to classify the scenario and utilizes the Scale Invariant Feature Transform (SIFT) algorithm [14] to analyze the motion status of the client. Based on the estimated video content, DeVA can adaptively select suitable configurations during runtime.

How to capture RoIs efficiently? Our observation in Section II-B shows that, unlike object detection, the error of depth estimation is affected by the entire frame. To balance the accuracy and bandwidth overhead, we define the background areas (whose depth values are high) as RoIs by analyzing the impact of different areas on the error. Moreover, considering RoIs usually contain elements not included in the object detection dataset, such as sky and wall, conventional object tracking approaches are inefficient. DeVA proposes an RoI capturing approach and a tracking approach to estimate RoIs on the device. To be specific, after obtaining the depth image and determining the RoI areas, the edge server performs connected component analysis and contour capturing, and calculates the minimum enclosing rectangle and inscribed rectangle to create bounding boxes. Since the RoI areas are usually covered by foreground objects, DeVA utilizes a lightweight object detection model to identify the foreground objects that cover the contour of RoIs. Based on these results, the client can capture the RoI more accurately by tracking RoI areas and foreground objects, respectively.

How to find the optimal configuration efficiently? Due to the extensive configuration space and diverse video content as well as RoI areas, selecting the optimal configuration online is prohibitively expensive. DeVA presents a profiling-based configuration adaptation approach to determine the resolution of the offloaded video and the QPs for all macroblocks. In the offline phase, DeVA simplifies configuration determination by classifying all macroblocks into RoI and non-RoI categories. To reduce the profiling overhead, DeVA utilizes the area ratio of RoI to non-RoI to identify the variation in RoI, which allows us to capture the configuration-RMSE relationship according to this ratio rather than for every possible RoI area. Based on the observation that the impact of configurations on error is largely independent, DeVA further minimizes the profiling cost by filtering out the configuration where the error exceeds a predefined threshold. Additionally, DeVA profiles different video content types to ensure the robust performance of the profiling-based configuration adaptation. At runtime, DeVA selects the optimal configuration according to the profiling, aiming to minimize the video size while maintaining the error within acceptable thresholds.

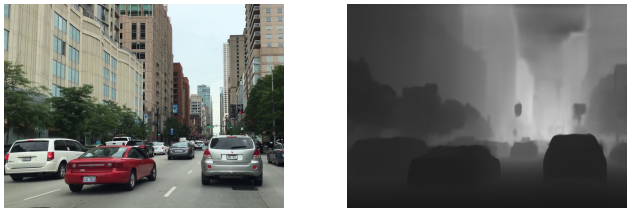

(a) Raw image (b) Depth image

Fig. 1. An example of depth estimation.

In summary, this paper achieves the following contributions:

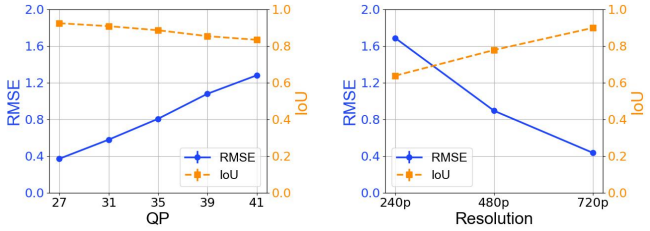
- We present DeVA, an edge-assisted video analytics framework for depth estimation that achieves high accuracy and low bandwidth overhead.
- We analyze the relationship between video analytics configurations and the error of depth estimation, and its key impact factors.
- We analyze the RoI for depth estimation, alongside an RoI tracking approach with low resource overhead.
- We propose a profiling-based configuration adaptation approach that reduces the bandwidth overhead while ensuring accuracy.
- We implement DeVA and demonstrate through evaluations that DeVA reduces the average RMSE by up to 47.48% compared to the baselines, and effectively keeps the error within the acceptable threshold. Additionally, DeVA reduces the average bandwidth overhead by 57.12% compared to raw-video transmission.

The rest of this paper is organized as follows. Section II introduces the background, defines the RoI area for depth estimation, and provides motivations for system design. Section III and Section IV present the system overview and the detailed design, respectively. Section V describes the implementation of DeVA. Section VI evaluates the performance of DeVA and Section VII summarizes the related work. This work is concluded in Section VIII.

II. Background and Motivation

A. Depth Estimation

Figure 1 illustrates an example of depth estimation. Distinctly different from the output of object detection, the depth image is typically grayscale where darker parts represent closer objects and lighter parts indicate objects further away. It is worth noting that each pixel in the depth image has a depth value representing the distance from the camera to the object. This feature implies that the RoIs for depth estimation, which significantly affects the performance of the depth estimation model, often contain elements not typically



(a) QP vs. RMSE (b) Resolution vs. RMSE

Fig. 2. The relationship between configurations and RMSE.

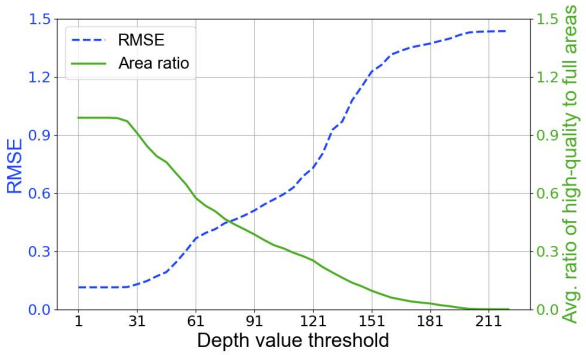


Fig. 3. The relationship among the depth value threshold, average RMSE, and the ratio of high-quality to total area. A lower quality setting (QP = 40) is applied to pixels with depth values below the threshold, while pixels above the threshold are encoded in high quality. The area ratio indicates the proportion of the frame that is preserved in high quality (i.e., depth values above the threshold) relative to the total area.

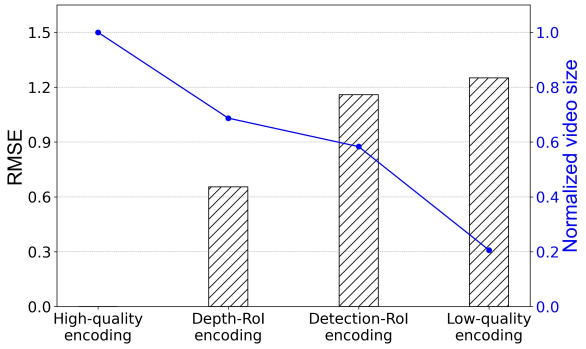


Fig. 4. RMSE and video size (normalized to the raw-video size) of different encoding methods.

targeted by common object detection classes. This point will be further discussed in the following Section II-B. As a result, detection-RoI encoding approaches are inefficient for capturing RoIs specific to depth estimation.

Based on the type of input image, depth estimation can be classified into monocular depth estimation and stereo depth estimation. Unlike the stereo depth estimation which requires multiple synchronized cameras positioned at a fixed distance apart, monocular depth estimation can measure the depth from a single image. RL-based monocular depth estimation, like ZoeDepth [15] and Depth Anything [16] can achieve this without any professional equipment (e.g., RGB-D cameras [17] and lidar [18]) or auxiliary information, making it more accessible and widely applicable.

Existing work generally employs several metrics to measure the error of depth estimation. In this work, we primarily utilize the RMSE which indicates the average difference between the ground truth and the estimation value. The RMSE is calculated by $\text{RMSE} = \sqrt{\frac{\sum_{i \in N} ||d_i - d_i^*||^2}{|N|}}$, where d_i and d_i^* are the estimated depth value and ground truth of pixel i, respectively. Other commonly used metrics include absolute relative error (REL) $= \frac{\sum_{i \in N} \frac{d_i - d_i^*}{d_i^*}}{|N|}$, and threshold accuracy: $\%$ of d_i, s.t., $\max(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}) = \delta < thr$.

B. Motivation

Limitations of Existing Approaches. Edge-assisted video analytics is a promising approach for ensuring video analytics accuracy, which, however, mostly focuses on "discrete" video analytics tasks and often neglects depth estimation. Specifically, coarse-grained configuration adaptation frameworks, such as Chameleon [6] and DARE [7], adapt the system configurations based on the profiling of the relationship between system configurations and detection accuracy. We evaluate the error of depth estimation and object detection under different resolution and QPs (the evaluation setting is consistent with that in Section VI-A). The evaluation results in Figure 2 show that video compression by reducing the QP and resolution indeed hurts the accuracy of object detection, but increases the error of depth estimation more significantly: As shown in Figure 2a, RMSE of depth estimation increases by 2.47 times when the QP varies from 27 to 41, while IoU of object detection decreases by only 9%. We also evaluate the use of super-resolution techniques for depth estimation. However, the results indicate that the additional time overhead introduced by super-resolution is prohibitive. Specifically, upscaling a single image from 270p (480×270) to 1080p (1920×1080) using recent super-resolution algorithms, i.e., SinSR [19] and ResShift [20], takes 2.22 seconds and 3.64 seconds, respectively. Given this significant latency, we conclude that super-resolution is not suitable for edge-assisted depth estimation.

RoIs for Depth Estimation. To balance the accuracy and resource consumption, some approaches dig into the RoI for object detection tasks and encode macroblocks of video frames with different quality. For example, EdgeDuet encodes small objects with high quality to ensure inference accuracy [10]. AccMPEG presents a lightweight model for the client to adaptively select the QPs for each macroblock [11]. RegionFilter presents a server-assisted approach for resource-limited edge nodes to efficiently encode and transmit RoIs in video frames [21]. Although these detection-RoI encoding approaches achieve high efficiency, they are impractical for depth estimation tasks since RoIs for depth estimation are not consistent with those for object detection. In the following, we will define the RoI for depth estimation and illustrate the need for designing RoI encoding in such tasks.

We first analyze the RoI for depth estimation, focusing on areas where the error of depth estimation is more sensitive to the encoding quality. We normalize the depth value of each video frame to a range of 0 to 255 (i.e., the range of one

color channel). We then simulate progressive degradation by incrementally applying low-quality encoding (with a quantization parameter, QP $= 40$) from the nearest scene depth (depth value = 0) to the farthest (depth value = 255). Specifically, we vary the depth value threshold from 0 to 255, such that all pixels with depth values below the current threshold are encoded with low quality, while those above the threshold retain high-quality encoding. The results in Figure 3 reveal that ***the background area plays a critical role in determining the accuracy of depth estimation***. For example, when the threshold is set to 91, approximately 39% of the frame is considered "background" (i.e., depth above the threshold) and encoded in high quality, while the remaining 61% is "foreground" and encoded in low quality. At this setting, the resulting RMSE remains relatively low, indicating acceptable estimation performance. However, as the threshold increases to 151, resulting in a substantial reduction of the high-quality background region to less than 10%, the RMSE increases sharply, exceeding 1.2. This suggests that overly compressing the background significantly degrades the overall accuracy of depth estimation. Based on these findings, ***we define the high-depth background region as the RoI for depth estimation*** and prioritize its encoding quality.

Figure 4 illustrates the error of depth estimation obtained by raw video, depth-RoI encoding, detection-RoI encoding, and coarse-grained low-quality encoding. The first one is to utilize the raw video for depth estimation and the last one is to encode the video with low quality (QP $= 40$). For the depth-RoI (or detection-RoI) encoding, we apply high quality (QP $= 23$) and low quality (QP $= 40$) to depth-RoI (or detection-RoI) areas and non-depth-RoI (or non-detection-RoI) areas, respectively. Specifically, the depth-RoI regions are determined using our method described in Section IV-C, and the detection-RoI regions are generated by YOLOv8 [22]. And other implementation details are consistent with Section V and Section VI-A. We find that detection-RoI encoding's error is only marginally lower than the coarse-grained approach. Compared with detection-RoI encoding, depth-RoI encoding reduces RMSE by 43.49%. The results identify that ***existing detection-RoI approaches are inefficient for the depth estimation task***. It is non-trivial to design an RoI encoding approach for depth estimation.

Impact of Video Content on Depth Estimation. To efficiently achieve the balance between the bandwidth overhead and accuracy, we should analyze the configuration-RMSE relationship and dig into the key impact factor on this relationship. The dataset and experiment setup are consistent with Section VI-A. The results in Figure 2 show that the error of depth estimation is more sensitive to the resolution: compared with QPs, the resolution drop significantly reduces the video size which is the key determinant of the video quality.

We further test the impact of video content on this configuration-RMSE relationship. Considering the usage scenario and behavior, we classify the video content based on the scenario (i.e., indoor and outdoor), period (i.e., day and night), and motion status of the client (i.e., moving or static). Results in Figure 5 illustrate that the depth estimation has better accuracy for the indoor scenario. The average RMSE outdoors can be 15.65 times worse than that indoors. This discrepancy may be due to the larger range of distances in outdoor settings, along with more complex elements and lighting conditions, which make depth estimation more challenging.

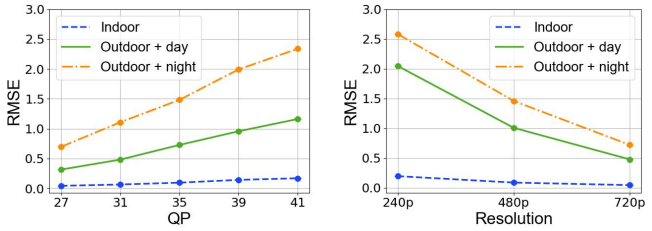
Figure 5 also compares the error of depth estimation during the daytime and at night. The results show that the depth estimation has poorer performance in dealing with the night scene. The error of depth estimation on the night scene can be at most 1.30 times higher than that on the day. We also find that the accuracy of depth estimation is more sensitive on the night scene: the RMSE increases more quickly when the quality gets worse. That is likely because poor lighting conditions at night not only increase noise in video frames but also reduce the effective range of vision.

Impact of Client's Motion Status on Depth Estimation. Figure 6 shows the impact of the client's motion status on the accuracy of depth estimation. The results indicate that the depth estimation performance is more sensitive when the client is moving, especially under low video quality. The average RMSE with a moving client is up to 24% higher than in the static case. That is because the aggressive compression strategy under high compression levels struggles to accommodate significant inter-frame variations in moving scenes, resulting in increased reconstruction errors.

III. System Overview

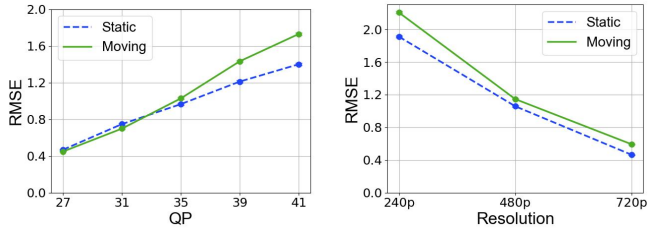
Figure 7 illustrates the system overview of DeVA. DeVA consists of seven components: Motion Analyzer, RoI Tracker, Depth Estimator, Content Analyzer, Optimizer, Encoder, and Decoder. The workflow is as follows: ❶ The video captured by the client is initially processed by the Motion Analyzer which estimates whether the client moves. The result will be sent to the Optimizer on the edge server for assistance in configuration determination. Then ❷ RoI Tracker filters keyframes and offloads them to the Depth Estimator on the edge server for depth estimation. ❸a Using the RoIs returned by the server, the tracker estimates the RoI areas for the remaining frames. ❹a These video frames and their corresponding RoIs are then passed to the Encoder.

Simultaneously, ❸b the Depth Estimator sends the RoI areas to the Optimizer. In addition to being processed by the Depth Estimator, ❸c the filtered frames are also passed to the Content Analyzer, which examines the video features and provides the analysis results to the Optimizer. Based on the RoI areas and the video content, ❹b the Optimizer determines the optimal configurations for the video clip, including the resolution and the QPs for each macroblock, to balance accuracy and bandwidth overhead. The Encoder then encodes the video clip according to this configuration, applying the specified QPs and resizing the resolution as needed. ❺ The encoded video is then transmitted to the edge server, where ❻ it is decoded and processed by the Depth Estimator to generate depth information for ❼ further processing such as AR content rendering and depth-based anomaly detection. The proposed framework ensures that DeVA efficiently balances the accuracy with network resource usage.



(a) QP vs. RMSE (b) Resolution vs. RMSE

Fig. 5. The relationship between configurations and RMSE under different video content.



(a) QP vs. RMSE (b) Resolution vs. RMSE

Fig. 6. The relationship between configurations and RMSE under different motion statuses of the client.

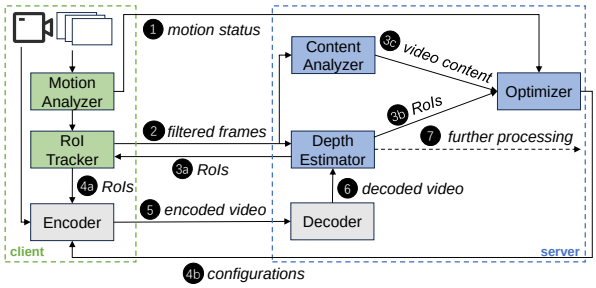


Fig. 7. System overview of DeVA.

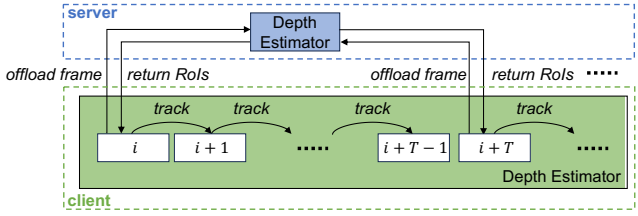


Fig. 8. Illustration of RoI Tracker.

IV. System Design

A. Motion Analyzer

As discussed in Section II-B, depth estimation is more challenging and less accurate when the client moves during the video capturing. Thus, it is essential to analyze whether the offloaded video is captured with a moving or static camera.

We use SIFT algorithm [14] to detect keypoints and compute descriptors of video frames, and find correspondences with a fast library for approximate nearest neighbors (FLANN) matcher. Then, the Motion Analyzer filters and evaluates those matches using the Lowe's ratio test to estimate if these consecutive frames are from the same scene or have undergone displacement, thus assessing the motion status of the client.

B. RoI Tracker

As the client has limited capacity, it is impractical to implement a depth estimation model on the client. We propose an RoI tracking approach that tracks RoI areas based on the RoIs of keyframes returned by the server. Figure 8 shows the workflow of RoI Tracker. The tracker will offload one video frame to the edge server for depth estimation and RoI analysis. Upon receiving the RoI results from the edge server, RoI Tracker will locally track the RoIs of the subsequent frames by following the motion of each RoI area.

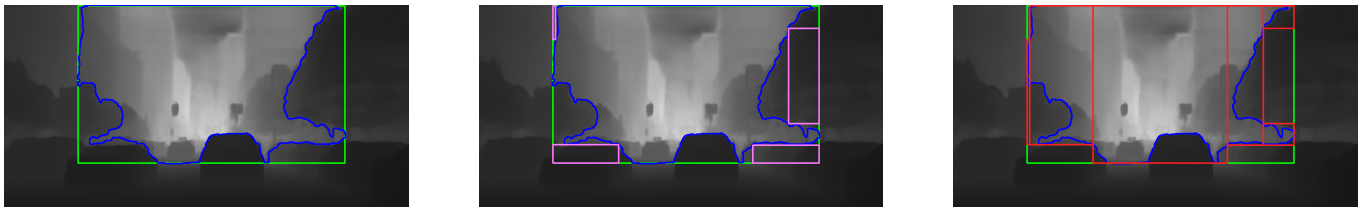
As the tracking algorithm is not perfect, the accuracy of tracking will decrease as the number of tracked frames increases [23]. To ensure the tracking accuracy, we first adopt the cosine similarity method to calculate the similarity between the feature vectors of the current tracking area and those of the previous frame's tracking area. By evaluating the level of feature matching between tracking areas in consecutive frames, we can effectively judge whether the tracked target has consistent moving trajectory. If the target is mismatched, the tracker will upload a new key frame to request the RoIs from the edge server, ensuring the accuracy and reliability of the tracking process.

As discussed in Section II-B, the RoI area for depth estimation often includes elements that cannot be recognized by the object detection model, rendering traditional tracking algorithms for object detection inefficient. By analyzing the RoI area in consecutive frames, we find that it is often covered by foreground objects. The variation of the RoI is not only due to its own movement but also because the occluded positions change as the foreground objects move. As a result, after estimating the RoI areas, the Depth Estimator further detects the objects covering the contour of the RoI, enabling the client to track the RoI area accurately. This detection process will be discussed in the following Section IV-C.

Based on our empirical observations, the RoI areas in outdoor scenarios are predominantly the background, such as the sky and wall. These areas typically remain stable over short periods. Consequently, the RoI Tracker will maintain fixed the RoI areas during the update period. For the foreground objects covering the contour of the RoI, we utilize the kernelized correlation filters (KCF) [24] to track the objects individually. KCF is a widely used tracking algorithm because of its high efficiency and robustness.

C. Depth Estimator

The Depth Estimator is responsible not only for conducting depth estimation but also for determining the RoI areas, creating the bounding boxes enclosing them, and detecting the foreground objects covering the contour of RoI.



(a) The contours (blue) and their bounding rectangles (green)

(b) The maximum inscribed rectangles of the contours (pink)

(c) The resulting bounding boxes (red)

Fig. 9. Illustration of creating bounding boxes for RoI.

As discussed in Section II-B, the depth image is grayscale and the RoI is the background area which affects the error of depth estimation significantly. Based on the observations in Figure 3, we define the threshold of depth value as the intersection point of average RMSE and area ratio. If the normalized depth value of a pixel in a depth image exceeds the threshold, we consider it belongs to RoI areas; otherwise, it is contained in non-RoI areas. As the error of depth estimation is affected by the full video frame, we believe this setting is a good balance between the performance and the bandwidth overhead. This is also verified by our evaluation.

To assist the RoI Tracker with efficient tracking, DeVA then create bounding boxes that enclose RoI areas. DeVA first performs connected component analysis on the depth image to identify the largest non-background (here the target RoI areas are regarded as foreground) rectangle that encompasses the RoI areas. DeVA then extracts the contours within this rectangle, and computes the bounding rectangle that encapsulates them. Figure 9a shows an illustration, the blue lines represent the contours of the RoI areas, and the green rectangles are the bounding boxes encapsulating these contours.

It is worth noting that if the contours are irregular, such as in concave regions, the bounding rectangle might contain irrelevant parts outside the actual region. For example, the left green rectangle in Figure 9a includes a large non-RoI area. To address this issue, DeVA refines the region by taking the intersection of the bounding rectangle and complement of the maximum inscribed rectangle (pink rectangles in Figure 9b). This refinement ensures the resulting region is more accurate and excludes extraneous areas. Finally, DeVA segments the refined region into multiple bounding boxes (depicted by the red rectangles in Figure 9c).

Given the bounding boxes enclosing RoI areas identified, the Depth Estimator then detects the objects covering them. To this end, DeVA passes the frame to an object detection model. Then it calculates the intersection of the RoI and the object detection results. The intersecting areas are classified as the covering areas. These results are finally returned to the RoI Tracker on the client for further processing.

D. Content Analyzer

As discussed in Section II-B, the video content, namely the scenario (indoor or outdoor) and period (day or night), significantly impacts the relationship between configurations and the error in depth estimation. This relationship, in turn, affects the configuration decision made by the Optimizer. To assist the Optimizer in making the optimal configuration decision for each macroblock, the Content Analyzer provides a method to identify the video content.

Scenario indicates the location where the user is. Different scenarios provide distinct ranges of visibility. For example, the length of a meeting room is typically about 10 meters, while users can see objects 50 meters away on a playground. As a result, the scenario affects both the accuracy of the depth estimation model as well as the accuracy required by users.

We classify the scenario as either indoor or outdoor based on specific features. DeVA utilizes ResNet18 [25], a lightweight deep neural network (DNN) architecture, to identify the scenario. To be specific, we train the scenario classification model using a public dataset from Kaggle [26] which consists of 348,512 images of indoor scenarios (e.g., shop, laundry, and living room) and 585,259 images of outdoor scenarios (e.g., highway, park, and mountain). We also collect outdoor-scenario videos from Bilibili (a user-generated content platform similar to YouTube in China) for training, yielding 13,779 extracted frames covering streets, cities, and other outdoor environments. To improve the accuracy of the scene identification, we utilize a queue to store the latest 5 classification results. We identify the scene with the most frequent classification result in the queue, unless the last 2 classification results are consistent.

Period refers to the time of day when the video is recorded. The levels of brightness vary significantly between day and night, affecting the difficulty of depth estimation. It is challenging for the depth estimation model to accurately capture depth values at night due to low visibility and reduced contrast.

To identify the period, we convert the video frame to a grayscale image and calculate the number of pixels that exceed a pre-defined threshold. We empirically set the threshold as 100 since the difference between day and night is pronounced at this value and it works well on all the videos in our dataset. When the frame contains a large number of pixels above this threshold, we classify the period as day; otherwise, it is classified as night. This method works since grayscale values are predominantly lower in nighttime images due to limited light, whereas daytime images have higher grayscale values due to abundant natural light, see Figure 10 for details. It allows the Content Analyzer to distinguish between day and night and identify the configuration-RMSE relationship accordingly. We also utilize a queue to store the latest period classifications. The period is identified based on the results in the queue, similar to the policy in scene identification.

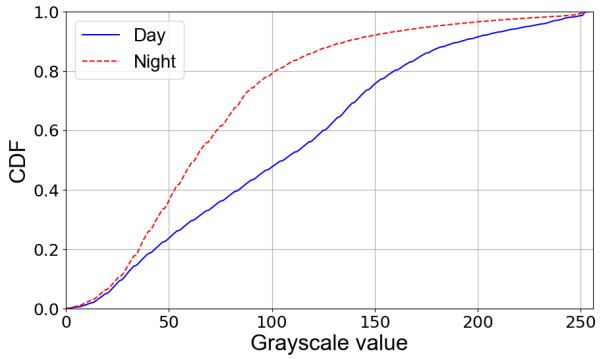


Fig. 10. The cumulative distribution function (CDF) of grayscale values under different periods, illustrating the probability that the grayscale value is less than or equal to a given threshold.

E. Optimizer

The Optimizer is to determine the configuration for each macroblock to ensure depth estimation accuracy and reduce bandwidth overhead as much as possible. Given the high number of macroblocks in a video frame (e.g., 8100 macroblocks in a 1080p video) and the large number of configuration candidates, it is prohibitively expensive to find the optimal configuration online. To address this challenge, we propose a profiling-based approach to enable efficient decision-making on the edge server. Using the profiling of the configuration-RMSE relationship collected offline, the Optimizer can decide on the configuration with a trivial time overhead.

Based on the observations in Section II-B, in the offline phase, we classify videos into two independent dimensions for the offline profiling phase: video content (indoor, outdoor+day, and outdoor+night) and the client's motion status (moving and static). This classification enhances the accuracy of the profiling-based approach. We propose four methods to reduce the profiling overhead.

Area-based QP assignment: As discussed previously, it is impractically costly to determine an optimal QP for each macroblock, especially for the high-definition video. To improve the time efficiency, instead of making fine-grained determination, we simplify the process by categorizing the video frame into two distinct areas: RoI and non-RoI. DeVA applies a high QP to the RoI area and a low QP to the non-RoI area. This method is aligned with state-of-the-art approaches for detection-RoI encoding [10].

Area ratio profiling: The RoI area varies with the object or camera movement, making it impractical to create configuration-RMSE profiling for every possible RoI area. To reduce the profiling overhead while maintaining performance, we utilize the area ratio of RoI to the non-RoI to represent the variation in RoI. This method allows us to effectively capture the RoI variation by profiling the configuration-RMSE relationship based on the area ratio of RoI to non-RoI.

Motion status identifying: Furthermore, as discussed in Section II-B, Depth estimation error becomes more pronounced when the client is moving, particularly in low video quality. It is necessary to slightly enhance the quality of non-RoI areas when the camera is moving. The evaluation results in Figure 6 indicate the effectiveness on ensuring the accuracy of depth estimation.

Resolution filtering: From the evaluation results in Section II-B, we find that the configuration-RMSE relationship is more sensitive to resolution and the impact of configurations on error is largely independent. Consequently, DeVA initially filters out resolutions where the error exceeds a predefined threshold. For instance, based on the results in Figure 5b, DeVA does not profile 240p and 480p resolutions in the outdoor+night scenario.

At runtime, the Optimizer queries the Motion Analyzer for the motion status of the client, the Content Analyzer for the video content and the Depth Estimator for the RoIs. Based on this information, the Optimizer can select the optimal configuration from the corresponding profiling and instruct the Encoder on the client for video encoding accordingly.

V. Implementation

In this section, we introduce the implementation of DeVA on the client and server, respectively.

Client. We use NVIDIA Jetson TX2 as the end device, as also used in related works [1], [2], [27]. We also utilize a Raspberry Pi 4B with 8GB memory to evaluate the overhead of DeVA on resource-limited device, as detailed in Section VI-E. We implement the functionality of video capturing and video splitting with OpenCV library [28]. The end device offloads the encoded video to the server using a TCP connection. As shown in Figure 7, DeVA contains three modules on the client side: Motion Analyzer, RoI Tracker, and Encoder. All these modules are performed on the CPU.

- Motion Analyzer: We use the SIFT_create method from OpenCV to detect keypoints and compute their descriptors for both images using the detectAndCompute function. Following this, we apply a FLANN-based matcher to find matches between the two sets of descriptors, filter these matches using the Lowe's ratio test [14], and evaluate the quality of the matches to judge if two consecutive frames are from the same scene or have undergone some displacement.
- RoI Tracker: We use KCF tracker [24] for detectable RoI tracking. When the tracking area mismatching, the Tracker will update a new key video frame.
- Encoder: The video encoder is implemented using open-source H264-QPBlock encoder [29]. This library is based on FFmpeg [30] and x264 [31], and supports encoding macroblocks with different target QP values. To support encoding with adaptive configurations, we modify the encoder and set the rate control mode as CRF, the adaptive quant mode as VARIANCE, and the default qp as 23 (consistent with the default setting of FFmpeg).

Server. We use a commodity server equipped with two 16-core CPUs, DRAMs of 128 GB in total, and two Nvidia RTX 4090 GPUs each with 24 GB memory. The modules of DeVA on the server side consist of four components: Content Analyzer, Depth Estimator, Optimizer, and Decoder. All modules run on the CPU, except for the Depth Estimator

and the scenario classification component of the Content Analyzer, which are executed on the GPU.

- Content Analyzer: We train the scenario classification model utilizing the ResNet18 [25] as the backbone model. The training dataset is from Kaggle [26] and videos collected from Bilibili. In the period classification approach, we utilize `cvtColor` in OpenCV to convert video frames to grayscale images, for measuring the brightness level of the video frame.
- Optimizer: The threshold of the depth estimation error is set to 1, as it is reported that an RMSE within this range is acceptable for practical users for AR applications [32]. The video analytics configurations consist of resolution in {480p, 720p, 1080p}, QPs for RoI areas in {23, 28, 33, 38}, and QPs for non-RoI areas in {25, 30, 35, 40}. The Optimizer also utilizes the H264-QPBlock encoder to capture the profiling of the configuration-RMSE relationship under the area-based QP assignment and camera movement.
- Decoder: DeVA utilizes `VideoCapture` module in OpenCV to decode the offloaded video for further depth estimation.
- Depth Estimator: The Depth Estimator uses ZoeDepth with two pre-trained weight configurations: ZoeD_K, trained on the KITTI dataset, and ZoeD_N, trained on the NYU dataset [15], to generate a depth map for each video frame. We set the threshold of normalized depth value as 75, 55, and 20 for outdoor + day, outdoor + night, and indoor contents, respectively, to balance the accuracy and bandwidth overhead of RoI encoding. To create the bounding boxes enclosing the depth image, we use OpenCV to analyze the connected component and extract contours, and utilize an open-source tool lir [33] to capture the maximum inscribed rectangle. For ease of RoI tracking, DeVA utilizes YOLOv8 with weights YOLOv8n [22] to identify the objects covering the contour of the RoI areas.

ZoeDepth provides different weights trained on the indoor dataset (NYU Depth v2 [34]) and the outdoor dataset (KITTI [35]). To ensure accuracy, we utilize the ZoeDepth model with pre-trained weights ZoeD_N for indoor scenarios and ZoeD_K for outdoor scenarios. To enable DeVA to adjust the depth estimation model, besides the process introduced in Section IV, the Content Analyzer also provides the estimated video content to both the Depth Estimator for creating RoIs and the Optimizer. The Optimizer then determines the appropriate model weights based on the video content. Upon receiving the video clip, the Depth Estimator will use the corresponding model for inference.

VI. Evaluation

A. Evaluation Setup

1) Video Datasets: We gather 30 video clips from YouTube which are captured by the dashboard camera, smartphone, or action camera. Each video clip ranges from 10 to 20 seconds in duration. All video clips in our study are 1080p (1920×1080) and 30 FPS. Our video dataset has diverse content, covering different periods (day and night) and scenarios (indoor and outdoor). To be specific, each category, i.e., outdoor + day, outdoor + night, and indoor, contains 10 videos.

2) Baselines: We use the following approaches as our baselines:

- GT, for which the client offloads the raw video to the edge server for depth estimation. Since the loss of human-annotated ground truth, we use the depth image given by GT as the ground truth to calculate RMSE.
- DetRoI, for which the system focuses on the object detection task and only takes RoIs for object detection into consideration. EdgeDuet [10] and AccMPEG [11] belong to this type of approach.
- DDS, for which the system utilizes two-round offloading to reduce the transmission overhead [9]. It first offloads a low-resolution video (e.g., 240p) and extract RoI for every frame to identify the feedback regions that need to be re-uploaded with higher quality.
- PtnDetRoI, a partition-based baseline inspired by AdaPyramid [36], divides each video frame into three vertical layers based on object detection (using YOLOv8l [22]). Objects are assigned to layers according to the centroid of their bounding boxes, and the maximal enclosing rectangle in each layer defines the RoI. A coarse-to-fine compression strategy is then applied from bottom to top.
- OptTrk, for which the system achieves the optimal RoI tracking accuracy. To be specific, the RoI area of every video frame is obtained using the method outlined in Figure 9.

B. Overall Performance

We first compare the overall performance of DeVA against four baselines under varying video contents. Figure 11a shows the comparison results on the average error of depth estimation. The results show that DeVA effectively keeps the RMSE within the specified threshold (RMSE $= 1$). As the GT approach offloads the raw video to the edge server for depth estimation and its resulting depth images are regarded as the ground truth, its error is 0. While GT achieves the best accuracy, as discussed later, its bandwidth overhead is the highest. As highlighted in Section II-B, the RoI significantly affecting depth estimation error differs from the RoI used for object detection. The evaluation results confirm this argument: the error obtained by the DetRoI approach increases by 41.72% compared with DeVA, and exceeds the RMSE threshold by 10.90%. For PtnDetRoI, dividing the frame into regions based on object detection and applying different quality levels proves ineffective in reducing RMSE: it yields an RMSE 32.10% higher than DeVA and improves on DetRoI by only 6.80%. For DDS, there is a significant 90.44% increase in error than ours and exceeds the RMSE threshold by 49.02%, indicating that it is not satisfactory to upload low-resolution videos to extract RoI for depth estimation. Finally, the OptTrk approach, with optimal accuracy in RoI tracking, achieves the lowest RMSE, but DeVA only induces an 3.67% RMSE increase compared to OptTrk, illustrating the high efficiency of the RoI Tracker.

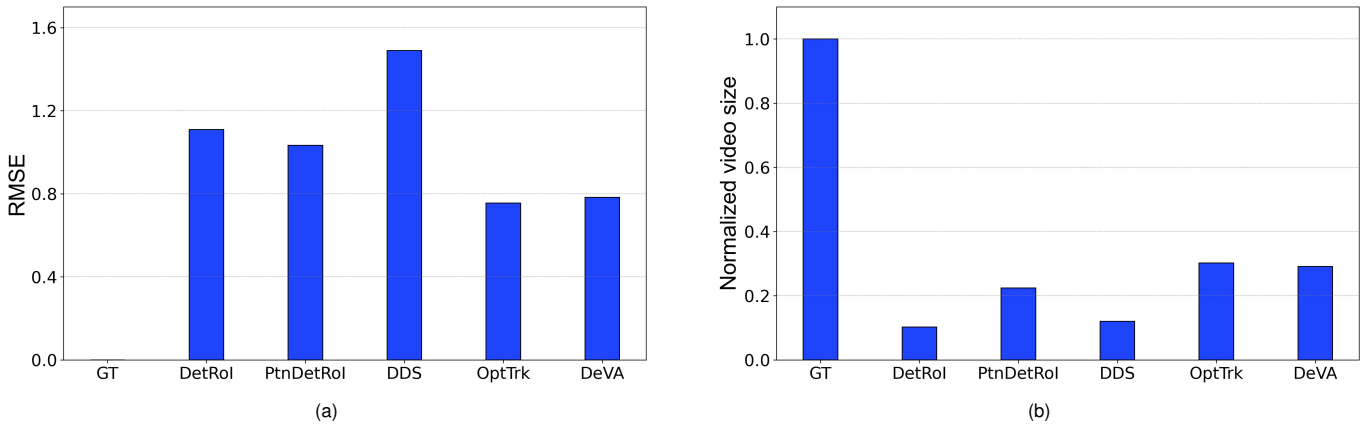


Fig. 11. Overall performance in (a) RMSE and (b) normalized video size.

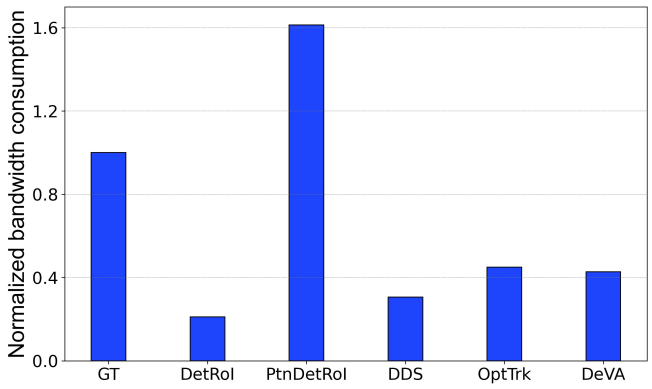


Fig. 12. The normalized bandwidth consumption of different methods.

Figure 11b compares the video sizes of all approaches, with sizes normalized to that of the raw video. The results show that DeVA, which adjusts the resolution and QPs for each macroblock, reduces videos size by 70.87% compared to GT. The result indicates that DeVA efficiently reduces the size of the uploaded video to save bandwidth. The DetRoI approach has minimal video size since it allocates large low-quality areas. As discussed in Section II-B, detection-RoIs typically involve detectable foreground objects and are smaller than the RoIs for depth estimation, which often include background areas. PtnDetRoI achieves a 22.97% smaller normalized video size compared to DeVA, but at the cost of significantly higher RMSE. This suggests that its partition-based approach is less effective than DeVA for depth estimation. It is worth noting that while DDS has a 58.84% smaller offloaded video size than DeVA, it results in a 90.44% increase in RMSE, indicating that DeVA can effectively reduce the error in depth estimation. As the optimal RoI is sometimes larger than the estimated one, the OptTrk approach results in a slightly higher video size compared with DeVA.

Figure 12 shows the bandwidth overhead of all approaches. The bandwidth consumption measures all the transmission overhead during the whole edge-assisted depth estimation process. Compared to GT, our method efficiently reduces the size of uploaded files by 57.12%, leading to substantial bandwidth savings. Although DetRoI and DDS achieve lower bandwidth consumption by 21.09% and 30.61%, they both fail to ensure the accuracy of depth estimation, indicating that DeVA strikes an acceptable balance between bandwidth overhead and accuracy. PtnDetRoI incurs a bandwidth overhead of 161.28%, significantly higher than the GT. This is because achieving layer-wise partitioning requires frequent uploads for server-side detection, including low-confidence regions. These regions are harder to track due to differences from the original PtnDetRoI method, resulting in more re-detections and thus higher bandwidth consumption. OptTrk causes a 5.77% higher bandwidth consumption than DeVA as it will make the configuration on resolution changes more frequently and have to offload a single frame rather than a compressed video when the resolution varies.

The above comparisons identify that DeVA achieves a great balance between the accuracy and bandwidth overhead among representative approaches.

C. Performance Breakdown

Figure 13 shows the performance under different video contents. Since the depth estimation generally has better accuracy for the indoor scenario, all approaches achieve good performance in limiting RMSE within the predefined threshold, as illustrated in Figure 13a. DeVA achieves a low error of depth estimation in outdoor scenarios, compared with DetRoI and PtnDetRoI. This is because the Content Analyzer of DeVA estimates the video content accurately and the Optimizer selects the configurations according to the estimated video content. It is worth noting that the RMSE of DeVA exceeds the predefined threshold by 5.59%. As the depth estimation model has the poorest performance in dealing with the night scene, the RMSE on outdoor + night scenarios is more sensitive to the accuracy loss of RoI tracking and configuration determination. However, DeVA's performance in this scenario is also better than the DetRoI, PtnDetRoI and DDS. As the DetRoI approach only takes the detectable foreground objects into consideration, it cannot provide acceptable performance under complex outdoor scenarios. Moreover, as the low resolution

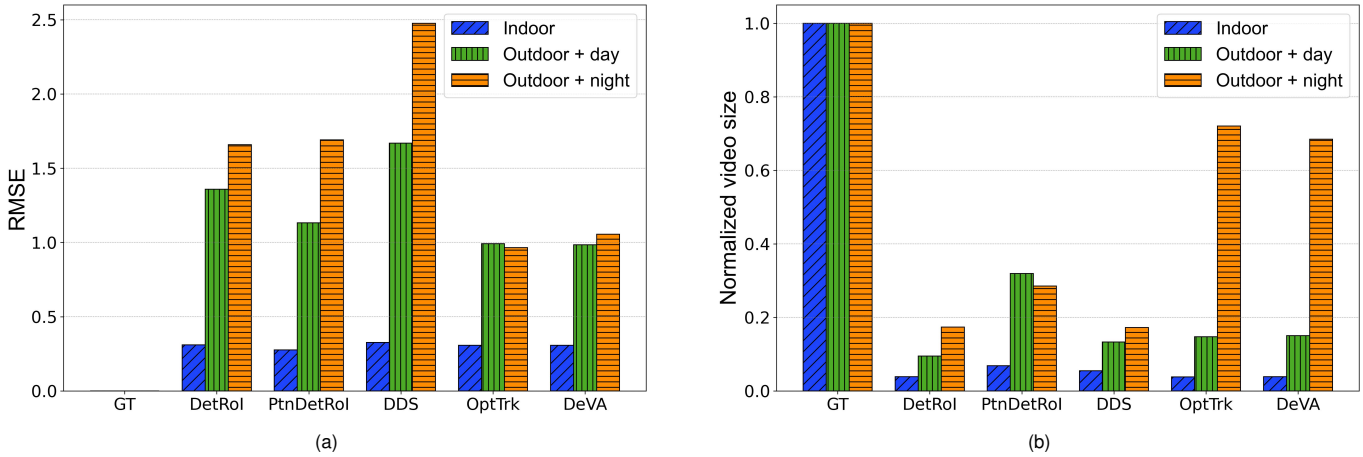


Fig. 13. Breakdown performance in (a) RMSE and (b) normalized video size under different video contents.

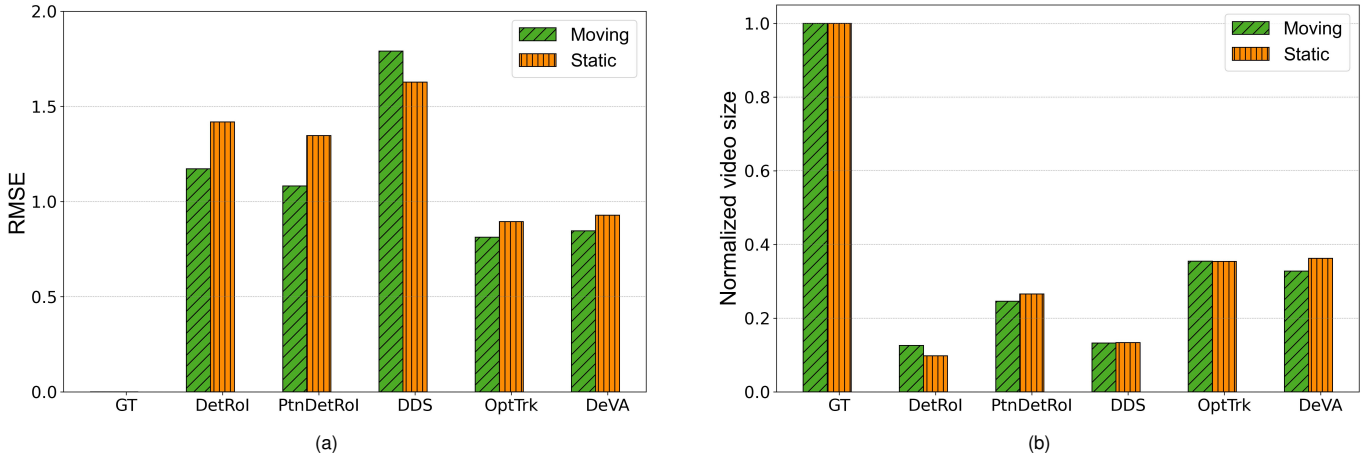


Fig. 14. Breakdown performance in (a) RMSE and (b) normalized video size under different motion status of the client.

will distort the depth value significantly, DDS cannot identify the RoI accurately in the first round, making its performance drops dramatically. Lastly, due to the reduced reliability of object detection in low-light conditions, particularly for objects near the top of the frame, PtnDetRoI may misallocate quality, assigning higher quality to less important regions and lower quality to critical ones, resulting in suboptimal RoI coverage and degraded depth estimation accuracy.

Figure 14 shows the performance under different motion statuses of the client. Our video dataset consists of 19 videos captured while in motion and 11 videos captured while stationary. DeVA achieves better accuracy under the RMSE threshold for both moving and static status. In moving status, DeVA achieves lower errors because it applies a slightly higher QP to non-RoI areas for the detail in Section IV-E. DetRoI exceeds the threshold by 17.17% in moving status and 41.77% in static status. PtnDetRoI surpasses the threshold by 8.20% during motion and 34.60% in static status. This difference in performance arises because object detection identifies fewer RoI areas in moving scenes, which enables the application of a lower QP to more non-ROI areas, thereby reducing overall error. Regardless of the motion status, DDS fails to meet the accuracy requirements for depth estimation tasks.

From the evaluation results in Figure 13b and Figure 14b, we find that the offloaded video sizes are distinct among different video contents. The reason is that as the relationship between configurations and RMSE is more sensitive in outdoor scenarios, the Optimizer has to choose a higher resolution and better QPs for both RoI and non-RoI areas to ensure the RMSE is within the threshold, leading to a higher video size. It is worth noting that the ratio of RoI area to total frame area is comparable between outdoor + day (51.56%) and outdoor + night (63.78%) scenarios, indicating that the RoI distribution is not the dominant factor. This further supports that the variation in video size is primarily attributed to the Optimizer's configuration choices in response to content sensitivity.

D. Accuracy of Scenario Classification

We further evaluate the performance of the ResNet18-based scenario classification model on both our dataset and the ACDC dataset [37]. The ACDC dataset contains diverse driving scenes under various environmental conditions, including challenging weather and lighting scenarios. As shown in Table I, the model achieves classification accuracies of

TABLE I
SCENARIO CLASSIFICATION ACCURACY ON OUR DATASET AND ACDC DATASET

Dataset	Scenario	Accuracy (%)
Ours	Outdoor	100.00
	Indoor	98.73
ACDC	Outdoor (extreme weather)	99.93
	Indoor	—

100% for outdoor scenes and 98.73% for indoor scenes, demonstrating its reliability in distinguishing general scenario. To assess its robustness under adverse conditions, we also test the model on extreme weather scenarios—night, fog, rain, and snow—using samples from the ACDC dataset. The model achieves an accuracy of 99.93%, confirming its effectiveness in handling a wide range of complex visual environments.

E. System Overhead

We further analyze the processing delay of DeVA. Figure 15 shows the average delay of key processes in DeVA at a bandwidth of 2Mbps. On the Jetson TX2, the primary device used in our evaluation, the tracking delay, i.e., the average processing time of the RoI Tracker on the client, is only 9.73 ms per frame. Although the RoI Tracker needs to wait for RoI areas captured and returned from the edge server, the time cost amortized over the video frames within one updating period is trivial, indicating good time efficiency. The encoding delay refers to the time required per frame to compress and encode, which is 73.81 ms. Motion analyzing delay refers to the time needed to assess the motion within the current video content, which is 789 ms. The majority of this time is spent on computing the matching of feature points between two consecutive frames to identify the client's motion status. The decision-making delay is the time between when the edge server receives the filtered video frame and when the Optimizer determines the configurations. Most of this delay is attributed to obtaining the depth image, which takes approximately 163 ms, and identifying RoIs, which takes about 439 ms. As the frequencies of motion analyzing and decision making are low (which are consistent with the RoI Tracker's updating period), these overheads are acceptable for practical use.

We further evaluate the overhead of DeVA on a Raspberry Pi 4B, as shown in Figure 15. While the overall latency increases compared to the TX2 platform, the most notable overhead comes from motion analysis, which rises by 8.6% to 857ms per calculation. Other components, including tracking and encoding, experience moderate increases that remain within an acceptable range. Despite the performance gap between platforms, the system still runs smoothly on the Raspberry Pi, demonstrating its feasibility on low-end edge devices.

DeVA imposes trivial system overhead at runtime. As discussed in Section IV-B, the client periodically sends video frames for updating reference RoI, and the edge server will return the RoIs and configurations to the client. The average network traffic for sending video frames is 64 KB/s, and the RoI and configurations exchanging is no more than 1 KB/s. On the edge server, DeVA consumes about 29.80% of the CPU and 4.78% of the memory on average. On the client, the RoI Tracker accounts for 2.50% of the CPU utilization and 4.00% of the memory utilization.

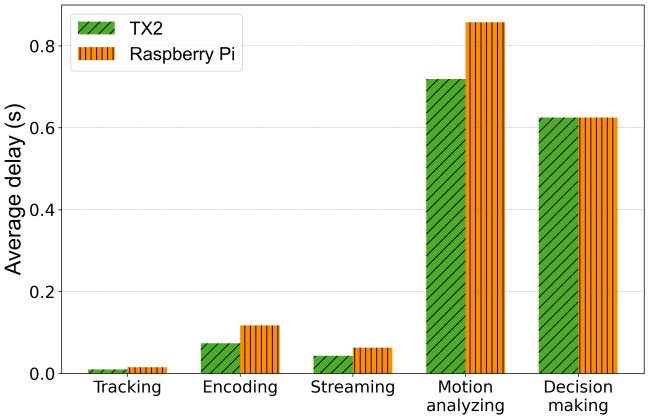


Fig. 15. The latency of key processes in DeVA on TX2 and Raspberry Pi 4B.

VII. RELATED WORK

This section reviews state-of-the-art depth estimation approaches and video analytics frameworks. Table II summarizes representative video analysis systems, comparing their tasks, architectures, core techniques, and objectives.

Depth estimation. Depth estimation is a critical vision task for AR, intelligent surveillance, autonomous driving, etc. Based on the input type, depth estimation can be classified into stereo depth estimation and monocular depth estimation. Stereo depth estimation utilizes the disparity between images captured from different viewpoints to estimate depth values. Zhang et al. propose a real-time and on-device depth estimation utilizing dual cameras [45]. Monocular depth estimation, which predicts depth from a single image, is more broadly applicable. Recent advancements leverage neural networks to enhance the accuracy of monocular depth estimation. Bhat et al. propose ZoeDepth, a model that predicts depth values by integrating relative and metric depth estimation techniques [15]. Luo et al. introduce an algorithm that fine-tunes the traditional depth estimation model to achieve high-accuracy and geometrically consistent results [3].

Edge-assisted video analytics. Edge-based video analytics has been extensively studied in recent years. Jiang et al. introduce Chameleon, which balances the tradeoff between resource consumption and accuracy by exploiting the spatial and temporary correlations in the video [6]. Kong et al. propose AccuMO, an edge-assisted multi-task scheduling framework that optimizes the accuracy of multiple augmented reality (AR) tasks, such as depth estimation and odometry, by scheduling offloaded frames and local trackers [38]. Although AccuMO considers the depth estimation task, it does not address the fine-coarse relationship between configurations and the accuracy of AR tasks. It also overlooks the analysis of RoI for depth estimation, leaving significant room for optimization in this area. Xiao et al. propose Yoda, a benchmark for video analytics to provide performance clarity [39]. Yoda

TABLE II
SUMMARY OF REPRESENTATIVE RELATED WORK.

Works	Tasks	System Architecture	Core Techniques	Objectives
DeVA	**Depth estimation**	**Client-server**	**Depth-RoI-based encoding and configurations adaptation**	**Ensure accuracy and reduce network resource overhead**
Chameleon [6]	Video analytics pipelines (VAPs)	Client-only	Configurations adaptation	Achieve higher accuracy or reduce resource consumption
AccuMO [38]	Depth estimation and odometry	Client-server	Dynamic scheduling multiple tasks	Improve overall task accuracy
Yoda [39]	Object detection	—	VAP performance clarity	Improve VAP evaluation
Wang et al. [40]	Semantic segmentation	Client-server	Super-resolution	Reduce bandwidth consumption
AccDecoder [41]	Object detection	Server-only	Adaptive frame selection, super-resolution, and inference result reuse	Improve accuracy and reduce latency
Mi et al. [42]	Object detection	Server-only	Resolution-involved Markov decision process	Improve accuracy and reduce latency
Dai et al. [43]	Object classification	Client-server	Collaborative inference	Reduce computational cost and transmitted data
EdgeDuet [10]	Object detection	Client-server	Tile-level parallelism	Improve accuracy and reduce latency
Chen et al. [44]	Image classification and object detection	Client-server	Contextualized image compression	Reduce bandwidth consumption and speed up
DDS [9]	Object detection and semantic segmentation	Client-server	Server-side DNN feedback	Improve accuracy or reduce bandwidth consumption
Accmpeg [11]	Object detection, semantic segmentation, and keypoint detection	Client-server	Encoding quality at each macroblock	Reduce latency and ensure accuracy
Elf [2]	Instance segmentation, object classification, and pose estimation	Client-server	Parallel offloading	Accelerate inference up and save bandwidth
CrossVision [12]	Object detection	Client-server	Information redundancy	Reduce latency and improve accuracy
AdaPyramid [36]	Object detection	Client-only	Frame partitioning	Reduce latency and ensure accuracy

compares the performance of state-of-the-art video analytics frameworks using proposed benchmark videos and provides a comprehensive understanding of their dependencies on video content characteristics.

Super-resolution imaging is a technique to reconstruct high-resolution images from low-resolution inputs. Utilizing this approach, Wang et al. propose to offload video streams in low resolution and reconstruct them to high-resolution frames on the cloud [40]. To further reduce the end-to-end latency, Yuan et al. propose using limited keyframes as input to the super-resolution model [41]. For the remaining frames, they present an approach that combines super-resolution transferring and inferencing result reuse to ensure inference accuracy. Building on this, Mi et al. [42] introduce a Markov Decision Process for adaptive resolution control, balancing inference accuracy and latency across varying video resolutions.

RoI-based video analytics. To further compress the video size, researchers focus on the RoI areas in each frame, encoding only the necessary parts with high quality. Dai et al. propose a collaborative inference approach that divides the object classification workload between the client and the server [43]. The client implements an extractor submodel to identify the RoI areas for object classification and offloads them to the server for performing classification. Wang et al. argue that the client can detect large-size objects, but it still struggles with small objects [10]. They propose to offload only small objects to the edge and use tile-level parallelism to reduce the end-to-end latency. Chen et al. propose a context-aware image compression optimization framework that identifies the importance of different image regions in a visual analytics task, enabling contextualized image compression for offloading [44].

Du et al. present a two-stage video analytics framework, DDS, for reducing the video streaming overhead: DDS sends a low-quality video stream to the server to identify areas requiring higher quality, and the client then re-encodes the video accordingly to enhance the inference accuracy [9]. To further reduce the end-to-end latency from the server-driven mechanism, Du et al. later propose a lightweight model for the client to select the QPs for each macroblock. They also utilize frame sampling and quality-assignment expansion to reduce the overhead [11]. Zhang et al. propose to offload the

RoI regions to multiple servers for parallel processing [2]. Zhang et al. proposes a distributed framework to match and balance the workload of RoIs between smart cameras with overlapping fields of view, achieving localized processing of video data [12]. Shi et al. propose a method that partitions frames to select different detection model weights to improve the accuracy of object detection [36]. These approaches, however, all consider the "discrete" video analytics tasks and are difficult to extend to depth estimation tasks.

VIII. Conclusion and Future Work

We present DeVA, an edge-assisted video analytics framework for depth estimation that supports depth RoI encoding. We define the area that significantly impacts the error of depth estimation as the RoI for depth estimation. To capture the RoI areas efficiently, DeVA proposes a method for creating bounding boxes enclosing RoI areas on the edge server and a tracking approach to estimate RoI areas on the client. Additionally, DeVA also measures the relationship between the video analytics configuration and the error of depth estimation, and identifies the key role of video content in affecting this relationship. DeVA provides a methodology to adaptively adjust the configurations, including the QPs for RoI and non-RoI areas and the resolution of encoded video, based on the video content, area ratio, and motion status of the client. Our evaluation results confirm that DeVA guarantees the accuracy of depth estimation while reducing the network resource overhead of video offloading.

Based on the insights and findings of this work, we identify several promising directions for future research: (1) to enhance the decision-making process by incorporating real-time constraints, enabling the system to adapt to dynamic bandwidth conditions and latency requirements; (2) to integrate adaptive bitrate streaming techniques as a secondary compression layer, achieving controllable accuracy in edge-assisted depth estimation under varying network conditions; (3) to expand the configuration space by incorporating frame rate and intra-frame refresh rate as tunable parameters, allowing more fine-grained control over the trade-off between bandwidth efficiency and analytics accuracy; and (4) to support multiple concurrent client requests by exploring scalable offloading strategies and improving server-side batching capabilities

References

[1] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *ACM MobiCom, Los Cabos, Mexico, October 21-25, 2019*. ACM, 2019, pp. 25:1–25:16.

[2] 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 *ACM MobiCom, New Orleans, Louisiana, USA, October 25-29, 2021*. ACM, 2021, pp. 201–214.

[3] X. Luo, J. Huang, R. Szeliski, K. Matzen, and J. Kopf, "Consistent video depth estimation," *ACM Transactions on Graphics*, vol. 39, no. 4, p. 71, 2020.

[4] T. Hu, H. Zhang, X. Zhu, J. Clunis, and G. Yang, "Depth sensor based human detection for indoor surveillance," *Future Generation Computer Systems*, vol. 88, pp. 540–551, 2018.

[5] G. M. Behara and V. P. Chodavarapu, "Towards autonomous depth perception for surveillance in real world environments," in *IEEE ICAC, Columbus, OH, USA, July 17-21, 2017*. IEEE Computer Society, 2017, pp. 77–78.

[6] J. Jiang, G. Ananthanarayanan, P. Bodík, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *ACM SIGCOMM, Budapest, Hungary, August 20-25, 2018*. ACM, 2018, pp. 253–266.

[7] Q. Liu and T. Han, "DARE: dynamic adaptive mobile augmented reality with edge computing," in *IEEE ICNP, Cambridge, UK, September 25-27, 2018*. IEEE Computer Society, 2018, pp. 1–11.

[8] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *IEEE INFOCOM, Honolulu, HI, USA, April 16-19, 2018*. IEEE, 2018, pp. 1421–1429.

[9] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang, "Server-driven video streaming for deep learning inference," in *ACM SIGCOMM, Virtual Event, USA, August 10-14, 2020*. ACM, 2020, pp. 557–570.

[10] X. Wang, Z. Yang, J. Wu, Y. Zhao, and Z. Zhou, "Edgeduet: Tiling small object detection for edge assisted autonomous mobile vision," in *IEEE INFOCOM, Vancouver, BC, Canada, May 10-13, 2021*. IEEE, 2021, pp. 1–10.

[11] K. Du, Q. Zhang, A. Arapin, H. Wang, Z. Xia, and J. Jiang, "Accmpeg: Optimizing video encoding for accurate video analytics," in *MLSys 2022, Santa Clara, CA, USA, August 29 - September 1, 2022*. mlsys.org, 2022.

[12] L. Zhang, Z. Lu, L. Song, and J. Xu, "Crossvision: Real-time on-camera video analysis via common roi load balancing," *IEEE Trans. Mob. Comput.*, vol. 23, no. 5, pp. 5027–5039, 2024. [Online]. Available: https://doi.org/10.1109/TMC.2023.3301391

[13] I. Ahmad, V. Swaminathan, A. Aved, and S. Khalid, "An overview of rate control techniques in HEVC and SHVC video encoding," *Multim. Tools Appl.*, vol. 81, no. 24, pp. 34 919–34 950, 2022. [Online]. Available: https://doi.org/10.1007/s11042-021-11249-5

[14] G. Tang, Z. Liu, and J. Xiong, "Distinctive image features from illumination and scale invariant keypoints," *Multim. Tools Appl.*, vol. 78, no. 16, pp. 23 415–23 442, 2019. [Online]. Available: https://doi.org/10.1007/s11042-019-7566-8

[15] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, "Zoedepth: Zero-shot transfer by combining relative and metric depth," 2023. [Online]. Available: https://arxiv.org/abs/2302.12288

[16] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth anything: Unleashing the power of large-scale unlabeled data," in *IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2024.

[17] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: real-time dense monocular SLAM with learned depth prediction," in *IEEE CVPR, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 6565–6574.

[18] K. Yoneda, H. T. Niknejad, T. Ogawa, N. Hukuyama, and S. Mita, "Lidar scan feature for localization with highly precise 3-d map," in *IEEE IV, Dearborn, MI, USA, June 8-11, 2014*. IEEE, 2014, pp. 1345–1350.

[19] Y. Wang, W. Yang, X. Chen, Y. Wang, L. Guo, L.-P. Chau, Z. Liu, Y. Qiao, A. C. Kot, and B. Wen, "Sinsr: Diffusion-based image super-resolution in a single step," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 25 796–25 805.

[20] Z. Yue, J. Wang, and C. C. Loy, "Resshift: Efficient diffusion model for image super-resolution by residual shifting," in *Advances in Neural Information Processing Systems*, vol. 36. Curran Associates, Inc., 2023, pp. 13 294–13 307.

[21] T. Chen, Y. Bu, Y. Zeng, L. Xie, and S. Lu, "Regionfilter: Region-aware video filtering mechanism on resource-constrained edge nodes," *Comput. Networks*, vol. 251, p. 110624, 2024. [Online]. Available: https://doi.org/10.1016/j.comnet.2024.110624

[22] G. Jocher, A. Chaurasia, and J. Qiu, "Yolov8," 2023, https://github.com/ultralytics/ultralytics.

[23] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *ACM SenSys, Seoul, South Korea, November 1-4, 2015*. ACM, 2015, pp. 155–168.

[24] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778.

[26] "Kaggle dataset: Indoor and outdoor," 2024, https://www.kaggle.com/datasets/mishayakovlev/indoor-outdoor.

[27] A. Zhang, C. Wang, B. Han, and F. Qian, "Yuzu: Neural-enhanced volumetric video streaming," in *USENIX NSDI, Renton, WA, USA, April 4-6, 2022*. USENIX Association, 2022, pp. 137–154.

[28] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2015.
[29] "H264-qpblock," 2022, https://github.com/Eynnzerr/h264_qpblock/.
[30] "Ffmpeg," 2024, https://ffmpeg.org/.
[31] "x264," 2024, https://code.videolan.org/videolan/x264.
[32] Y. Zhang, T. Scargill, A. Vaishnav, G. Premsankar, M. D. Francesco, and M. Gorlatova, "Indepth: Real-time depth inpainting for mobile augmented reality," *ACM IMWUT*, vol. 6, no. 1, pp. 37:1–37:25, 2022.
[33] "lir," 2023, https://github.com/OpenStitching/lir.
[34] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *ECCV, Florence, Italy, October 7-13, 2012, Proceedings, Part V*, vol. 7576. Springer, 2012, pp. 746–760.
[35] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *IEEE CVPR, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 3061–3070.
[36] X. Shi, S. Zhang, J. Wu, N. Chen, K. Cheng, Y. Liang, and S. Lu, "Adapyramid: Adaptive pyramid for accelerating high-resolution object detection on edge devices," *IEEE Transactions on Mobile Computing*, vol. 23, no. 8, pp. 8208–8224, 2024.
[37] C. Sakaridis, D. Dai, and L. Van Gool, "Acdc: The adverse conditions dataset with correspondences for semantic driving scene understanding," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 10 745–10 755.
[38] Z. J. Kong, Q. Xu, J. Meng, and Y. C. Hu, "Accumo: Accuracy-centric multitask offloading in edge-assisted mobile augmented reality," in *ACM MobiCom, Madrid, Spain, October 2-6, 2023*. ACM, 2023, pp. 30:1–30:16.
[39] Z. Xiao, Z. Xia, H. Zheng, B. Y. Zhao, and J. Jiang, "Towards performance clarity of edge video analytics," in *IEEE/ACM SEC 2021, San Jose, CA, USA, December 14-17, 2021*. IEEE, 2021, pp. 148–164.
[40] Y. Wang, W. Wang, J. Zhang, J. Jiang, and K. Chen, "Bridging the edge-cloud barrier for real-time advanced vision analytics," in *USENIX HotCloud, Renton, WA, USA, July 8, 2019*. USENIX Association, 2019.
[41] T. Yuan, L. Mi, W. Wang, H. Dai, and X. Fu, "Accdecoder: Accelerated decoding for neural-enhanced video analytics," in *IEEE INFOCOM, New York City, NY, USA, May 17-20, 2023*. IEEE, 2023, pp. 1–10.
[42] L. Mi, T. Yuan, W. Wang, H. Dai, L. Sun, J. Zheng, G. Chen, and X. Fu, "Accelerated neural enhancement for video analytics with video quality adaptation," *IEEE/ACM Transactions on Networking*, vol. 32, no. 4, pp. 3045–3060, 2024.
[43] X. Dai, X. Kong, T. Guo, and Y. Huang, "Cinet: Redesigning deep neural networks for efficient mobile-cloud collaborative inference," in *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*. SIAM, 2021, pp. 459–467.
[44] B. Chen, Z. Yan, and K. Nahrstedt, "Context-aware optimization for bandwidth-efficient image analytics offloading," *ACM Trans. Multim. Comput. Commun. Appl.*, vol. 20, no. 9, pp. 262:1–262:22, 2024. [Online]. Available: https://doi.org/10.1145/3638768
[45] J. Zhang, H. Yang, J. Ren, D. Zhang, B. He, T. Cao, Y. Li, Y. Zhang, and Y. Liu, "Mobidepth: real-time depth estimation using on-device dual cameras," in *ACM MobiCom, Sydney, NSW, Australia, October 17 - 21, 2022*. ACM, 2022, pp. 528–541.

Shutong Chen is an Assistant Professor at the School of Computer, Electronics and Information at Guangxi University. She received her Ph.D. in computer architecture from the School of Computer Science and Technology, Huazhong University of Science and Technology, and received her B.Sc. degree from the School of Mathematics, Hunan University. Her research interests include edge computing and green computing.

Jingwen Yin received the BE degree from the Nanjing University of Finance & Economics, China, in 2023. He is currently working toward the Master degree with the School of Computer, Electronics and Information at Guangxi University. His research interests include edge computing.

Ruichao Zhong received the BE degree from the Lanzhou Jiaotong University, China, in 2017. He is currently working toward the Master degree with the School of Computer, Electronics and Information at Guangxi University. His research interests include edge computing and privacy protection.

Fangming Liu is currently a Full Professor with the Huazhong University of Science and Technology, Wuhan, China. His research interests include cloud/edge computing, datacenter and green computing, SDN/NFV/5G and applied ML/AI. He received the National Natural Science Fund (NSFC) for Excellent Young Scholars, and the National Program Special Support for Top-Notch Young Professionals. He is a recipient of the Best Paper Award of IEEE/ACM IWQoS 2019, ACM e-Energy 2018 and IEEE GLOBECOM 2011, the First Class Prize of Natural Science of Ministry of Education in China, as well as the Second Class Prize of National Natural Science Award in China.