2025 IEEE 45th International Conference on Distributed Computing Systems (ICDCS) | 979-8-3315-1723-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICDCS63083.2025.00032

DiVE: Differential Video Encoding for Online Edge-assisted Video Analytics on Mobile Agents

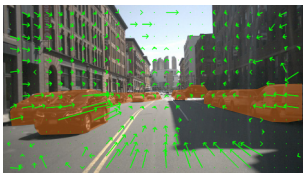
Jiangang Shen*, Hongzi Zhu*§, Liang Zhang†, Yunzhe Li*, Shan Chang†, Jie Wu‡, and Minyi Guo*
*Shanghai Jiao Tong University, China
†Donghua University, China
‡Cloud Computing Research Institute China Telecom, China
{sjg19970410, hongzi, yunzhe.li, myguo}@sjtu.edu.cn
{zhangliang, changshan}@dhu.edu.cn
wujie@chinatelecom.cn

***Abstract*—Ensuring stable and high-quality real-time video analytics for computationally constrained mobile agents is essential. However, limited computing resources and network bandwidth present significant challenges in meeting the objective of low response time and high inference accuracy. In this paper, we present DiVE, an edge-assisted video analytics system that utilizes motion vectors calculated by video codec to extract foregrounds and differentially encode frames. DiVE removes rotational components from motion vectors by solving over-determined linear equations and filters noisy motion vectors based on the observation that motion vectors of static objects point to the same point when the ego agent purely translates. To distinguish foregrounds from backgrounds, DiVE estimates the ground based on observations that all foregrounds stand on the ground and motion vectors on static objects at the same height have the same normalized magnitude. DiVE then uses region-growing-based clustering to identify foreground objects. An adaptive bitrate allocation method is applied to optimize accuracy under estimated bandwidth. We conduct extensive experiments to evaluate the performance of DiVE. The results demonstrate that DiVE can improve detection accuracy by up to 39.1% and reduce response time by up to 19.1% compared with other video analytics schemes on nuScenes and RobotCar datasets.**

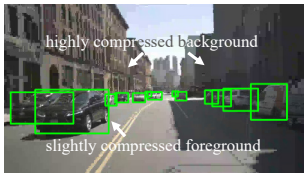
***Index Terms*—mobile agents, motion vector, video streaming, video analytics, deep learning**

I. Introduction

Recent years have witnessed a rapid increase of various types of mobile agents, such as autonomous driving agent on a normal vehicle, embodied AI agent on a robot, and smart agent on a drone, which may exist as an independent module and feature their hosts with many appealing intelligent mobile applications. Among many of the others, live video streams generated from mobile agents need fast treatment. For example, autonomous driving requires the system to have a comprehensive understanding of the scene and all related objects as quickly as possible in the real world. However, most mobile agents can only conduct limited computation [1]. Therefore, computer vision tasks such as object detection, classification and semantic segmentation need to be accomplished nearly in real time for fast video analytics by offloading most computational workload to an edge server.



(a) Foreground can be estimated using calculated motion vectors



(b) Accurate object detection on a differentially encoded frame

Fig. 1: An illustration of object detection on a DiVE-encoded video, where (a) the region of foreground objects in each frame of the video can be well estimated, using motion vectors obtained from a traditional video codec; and (b) each frame is differentially encoded with the foreground remaining clear and sharp yet the background heavily blurred, so that the bitrate of the live video stream adaptively fits the uplink bandwidth without affecting the final object detection accuracy.

However, an effective edge-assisted online video analytics system needs to meet three rigid requirements. First, the system should achieve high analysis accuracy as inaccurate results may lead to severe consequences, such as car crashes. Second, the system should have a low response time even with a highly unstable wireless communication connection, especially for fast moving agents. Last but not least, the system should be lightweight and scalable given the limited computational power of a mobile agent and the potential huge number of agents in the system.

In the literature, existing edge-assisted video analytics schemes can be classed into three categories, *i.e.*, frame filtering at camera side, video compressing based on server feedback, and local compressed model assisted video analytics. The frame filtering methods [2]–[4] adopt simple logic to recognize key frames and send them to the edge for model inference. These methods perform poorly in highly dynamic scenarios due to the detection results of other regular frames rely heavily on tracking the bounding box of key frame. Existing compression-based methods [5]–[7] allocate more bits to regions of a video frame that the edge deems to

§Corresponding author

2575-8411/25/$31.00 ©2025 IEEE
DOI 10.1109/ICDCS63083.2025.00032

significantly impact inference accuracy, while reducing the transmission bits for the remaining regions. However, these methods depend on guidance from the edge, leading to huge latency. The last category of video analytics system rely on light-weight models [8]–[10] deployed on mobile devices to handle simple sub-tasks like large object detection or appearance of target objects. These methods inherently require significant computing power and demonstrate diminished accuracy when processing complex video content. Other studies [11], [12] adjust video encoding configurations like resolution or frame rate to chase maximum inference accuracy based on current available bandwidth. Nevertheless, such methods require offline profiling and cannot handle live video streams.

In this paper, we propose DiVE, a novel differential video encoding scheme for online edge-assisted video analytics on mobile agents. The core idea of DiVE is to leverage the low-cost motion vectors generated by video codec to extract foreground regions of interest so that the best image quality is reserved for the foreground while the bitrate of the final encoded video fits the current uplink bandwidth. As a result, DiVE only conducts limited computation for basic analytics but can achieve low response time and high video analytics accuracy at the same time. Figure 1 illustrates an example of object detection on a DiVE-encoded video where each frame is differentially encoded with the foreground remaining clear and sharp yet the background heavily blurred without affecting the final object detection accuracy.

There are two main challenges in designing DiVE. First, although motion vectors have distinct features for identifying different objects, they are very coarse and vulnerable to image noise and rotation of the camera, which makes it hard to use. To tackle this challenge, we conduct extensive empirical studies and have the key observation that when the agent only translates in space, all motion vectors on static objects point to the same point, referred to as focus of expansion (FOE), which be well utilized to filter out noise motion vectors. Furthermore, when the agent moves forward, its pitch and yaw angles can be well estimated by solving over-determined linear equations with carefully selected motion vectors. As a result, the rotational component can be effectively removed from a motion vector caused by compound motion.

Second, it is difficult to distinguish the foreground regions from the background with only motion vectors as foreground objects may have distinct motion and different image textures. We have the following two key observations: 1) all foreground objects stand on the ground and 2) non-rotational motion vectors on objects of the same height (*e.g.*, the ground) in the real world have the same normalized magnitude. To deal with this challenge, we first estimate the ground region based on the above observations. Then, we propose a region-growing-based clustering algorithm to identify foreground objects, starting from those object seeds standing within the ground region. Finally, convex hulls of similar objects can be generated, forming multiple foreground regions.

The DiVE agent is lightweight and written in C++. We conduct extensive experiments on two public driving video datasets, *i.e.*, RobotCar and nuScenes. Results demonstrate that DiVE can achieve high inference accuracy under various bandwidth settings.DiVE can achieve a large accuracy gain of up to 39.1% and reduce 19.1% response time, compared with the state-of-the-art edge-assisted video analytics methods. We highlight the main contributions made in this paper as follows: 1) a lightweight motion vector based foreground extraction scheme is proposed 2) an adaptive video encoding scheme is proposed 3) extensive experiments are conducted, demonstrating the efficacy of DiVE.

II. Problem Definition and Preliminaries

A. System Model and Problem Definition

We consider two types of entities in an edge-assisted video analytics system:

- **Mobile Agents:** A mobile agent, *e.g.*, a smart dashcam mounted on the windshield of a vehicle, is capable of generating live video streams while demanding video analytics such as object detection in real time. The mobile agent has constrained computational power but can run basic video encoding operation such as H.264 or MPEG-4. In addition, the agent can exchange information with an edge server via 4G/5G mobile communication networks. We consider that mobile agents may move in a high speed, which causes the uplink bandwidth to dramatically change when uploading encoded video streams to an edge server for analytics.
- **Serverless Edge Computing:** Upon receiving video analytics requests, a unified serverless computing fabric (SCF) for the edge-cloud continuum has sufficient computational power for video decoding and online deep neural network (DNN) model inference, and returns the results to mobile agents with low end-to-end model inference and communication latency.

Given the dynamic uplink bandwidth, the online edge-assisted video analytics problem is defined as: how to successfully encode and transmit a live video stream at a mobile agent so that the edge server can successfully decode the received video and achieve the required analytics accuracy using existing pre-trained DNN models, while meeting the end-to-end real-time constraint at the same time.

B. Video Encoding

To compress a video, frames are organized into structures known as Groups of Pictures (GoPs). Each GoP begins with an *I*-frame, which is encoded independently without any reference to other frames. Following the *I*-frame are multiple *P*-frames and *B*-frames where only the differences from reference frames are encoded. Given a new frame and a reference frame, the encoder processes the new frame in unit of macroblocks (*e.g.*, the typical size of a macroblock is 16×16 pixels) in the following three steps. First, for each macroblock in the current frame, it searches for the most similar macroblock in the reference frame so that the amount of residual data needed for encoding is reduced. The displacement from the current macroblock to the corresponding macroblock in the

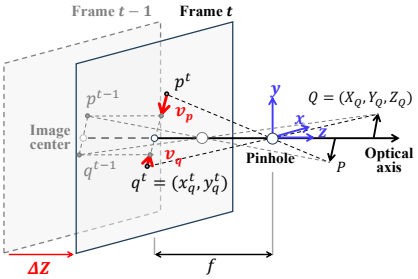


Fig. 2: Motion vectors of static objects caused by camera translation in z-axis.

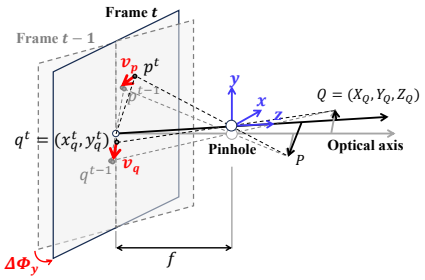
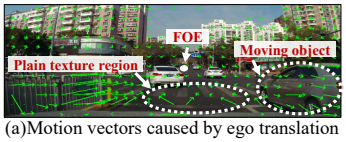


Fig. 3: Motion vectors of static objects caused by camera rotation in y-axis.



(a)Motion vectors caused by ego translation

(b) Motion vectors caused by a left turn

Fig. 4: Examples of calculated motion vectors.

reference frame is represented by a two-dimensional *motion vector*. Second, given the required bitrate and the differential encoding requirement, represented as a quantizer parameter (QP) offset map where a macroblock should be compressed harder would have a positive value, it determines the specific QP value for each macroblock. Finally, it performs entropy encoding on the transformed and quantized data [13].

C. Motion Vectors Caused by Camera Translation

We can characterize a general camera with a simple pinhole camera model. As illustrated in Figure 2, a static point Q in the physical world with coordinates (X_Q, Y_Q, Z_Q) in the world coordinate system is projected through the pinhole to a point q on the image plane of the camera with image coordinates (x_q^t, y_q^t) at frame t. If the camera coordinate system coincides with the world coordinate system, (X_Q, Y_Q, Z_Q) and (x_q^t, y_q^t) satisfies the equations as follows

$$x_q^t = f\frac{X_Q}{Z_Q}, \quad y_q^t = f\frac{Y_Q}{Z_Q}, \tag{1}$$

where f is the focal length represented in the units of pixels.

When the camera translates with a displacement of ΔZ along z-axis from frame $t-1$ to frame t, the projection point moves from q^{t-1} to q^t with a motion vector of v_q, which can be decomposed to vx_q and vy_q along the x-axis and y-axis, respectively, calculated as

$$vx_q = \frac{\Delta Z x_q^t}{Z_Q}, \quad vy_q = \frac{\Delta Z y_q^t}{Z_Q}. \tag{2}$$

In general, if the camera translates in an arbitrary direction in the physical world with the respective displacement of ΔX, ΔY and ΔZ along each axis, vx_q and vy_q can be calculated as

$$vx_q = \frac{\Delta Z}{Z_Q}(x_q^t - \frac{\Delta X f}{\Delta Z}), \quad vy_q = \frac{\Delta Z}{Z_Q}(y_q^t - \frac{\Delta Y f}{\Delta Z}). \tag{3}$$

We have the following observation:

Observation 1. *When the camera translates forward, motion vectors of static objects point to the same point, referred as focus of expansion (FOE), which also coincides with the vanishing point [14]. Moreover, the magnitude of these vectors are proportional to both the depth of corresponding objects and the distance between the projected points and FOE on the image plane.*

From Eq.(3), the position of FOE on the image plane is $(\frac{\Delta X f}{\Delta Z}, \frac{\Delta Y f}{\Delta Z})$. Figure 4(a) depicts an example of motion vectors calculated when a camera mounted on a vehicle moves forward. It can also be seen that Observation 1 does not hold for motion vectors calculated on moving objects. On one hand, these vectors are distinctive and can be utilized for identifying these moving objects. On the other hand, they are more complicated caused by the relative movement between these moving objects and the camera. In addition, motion vectors in regions with plain textures are hard to calculate and seem noisy.

D. Motion Vectors Caused by Camera Rotation

When the camera rotates $\Delta\phi_y$ around its y-axis as illustrated in Figure 3, vx_q and vy_q are calculated as

$$vx_q = -\Delta\phi_y f - \frac{\Delta\phi_y {x_q^t}^2}{f}, \quad vy_q = -\frac{\Delta\phi_y x_q^t y_q^t}{f}. \tag{4}$$

Generally, if the camera rotates around each axis with the respective degrees of $\Delta\phi_x$, $\Delta\phi_y$ and $\Delta\phi_z$, vx_q and vy_q can be calculated as

$$\begin{aligned} vx_q &= -\Delta\phi_y f + \Delta\phi_z y_q^t + \frac{\Delta\phi_x x_q^t y_q^t}{f} - \frac{\Delta\phi_y {x_q^t}^2}{f}, \\ vy_q &= \Delta\phi_x f - \Delta\phi_z x_q^t - \frac{\Delta\phi_y x_q^t y_q^t}{f} + \frac{\Delta\phi_x {y_q^t}^2}{f}. \end{aligned} \tag{5}$$

Figure 4(b) depicts an example of motion vectors calculated when the vehicle makes a left turn. It can be seen that rotation also violates the motion vector distribution described in Observation 1, which makes it hard to identify objects of interest.

E. Datasets

We consider three public mobile video datasets for study as follows:

1) KITTI: The *KITTI* [15] dataset is a widely used benchmark in autonomous driving research. The KITTI dataset is collected by driving in rural areas and on highways around Karlsruhe. The video data is captured with a frame rate of 10 frames per second (FPS) and image resolution is 1242×375 pixels. In the KITTI dataset, inertial measurement unit (IMU) data is sampled at 100Hz, including three-axis linear acceleration and three-axis angular velocity. The IMU data is

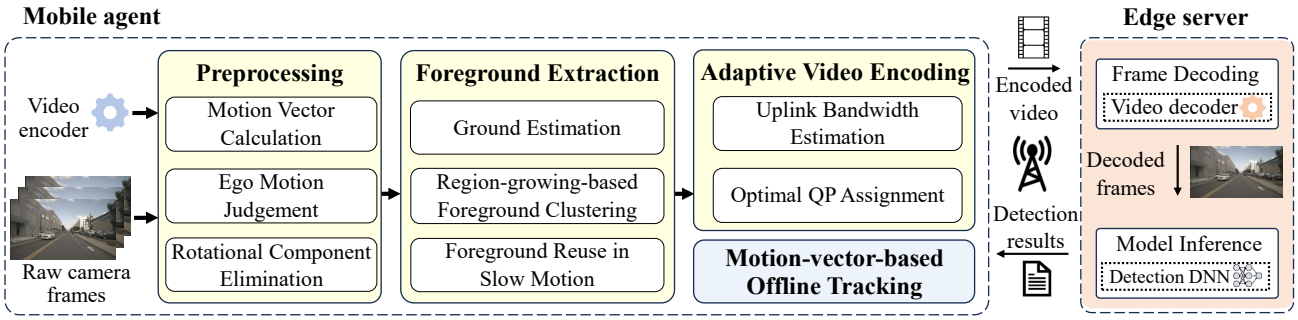


Fig. 5: Architecture of DiVE.

recorded with timestamps to enable precise synchronization with camera frames, ensuring temporal alignment between every frame and its corresponding IMU measurements.

2) nuScenes: The *nuScenes* [16] dataset is a public large-scale dataset for autonomous driving. The nuScenes dataset is collected in Singapore and Boston with a frame rate of 12 FPS, and image resolution is 1600×900 pixels. In the nuScenes dataset, every sixth video frame is sampled as a keyframe and annotated. The annotations include 3D bounding boxes *i.e.,* position, dimensions and orientation, and object categories *e.g.,* vehicles, pedestrians. The nuScenes test set is consisted of 150 video clips with each length of 20s under various traffic scenarios.

3) RobotCar: The *RobotCar* [17] dataset is an autonomous driving dataset collected in Oxford, UK over one year on a 10km route. The RobotCar dataset consists of approximately 1000km of recorded driving and video data is captured with a frame rate of 16 FPS under various weather conditions. In the RobotCar dataset, each video clip is labeled with weather conditions. The total number of frames is over 20 million and image resolution is 1280×960 pixels.

III. Design of DiVE

A. Overview

The core idea of DiVE is for a resource-constrained mobile agent to leverage motion vectors output by the video codec to identify foreground objects in each frame, and adaptively choose different compression ratios between the foreground and the rest of a frame, according to the current uplink bandwidth. As a result, the encoded video can best fit the dynamic uplink condition while maintaining supreme image quality for these regions of interest, achieving high model inference accuracy at the edge server and low end-to-end response time. Figure 5 depicts the architecture of DiVE, where a mobile agent uploads differentially encoded video to an edge server for analytics and the server returns detection results back to the agent for various downstream tasks. To this end, one mobile agent integrates four effective components:

Preprocessing: For each frame, given the raw motion vectors generated by the video codec, the motion status of the agent is estimated according to the distribution of motion vectors. The motion vectors are corrected by eliminating corresponding rotation components.

Foreground Extraction (FE): Given the fact that foreground objects stand on the ground and similar motion vectors can be observed on the same object, FE first utilizes a unique feature of motion vectors to neatly estimate the ground region in each frame. Then, objects are identified by clustering similar motion vectors starting from the ground region. Based on the detected motion status, if the agent comes to a stop when no motion vectors can be used for ground estimation, FE reuses the latest estimated foreground.

Adaptive Video Encoding (AVE): Given the identified foreground regions in each frame, AVE first measures the uplink bandwidth and then assigns different compression ratios, referred to as QP values, for the foreground regions and for the rest of the frame, respectively, resulting in an encoded video with detailed foregrounds that best fits the current uplink condition.

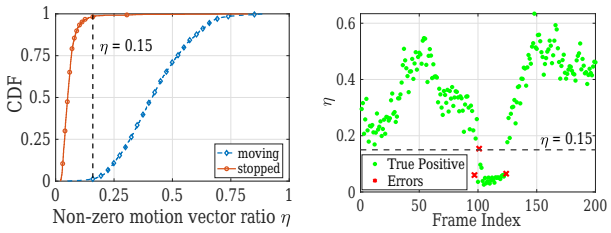
Motion-vector-based Offline Tracking (MOT): When an encoded frame cannot be transmitted to the edge server in time due to a severely impaired uplink, MOT conducts local object tracking based on motion vectors to derive the detection result of the frame.

B. Preprocessing

As analyzed in Section II, utilizing raw motion vectors has the potential to distinguish objects of interest in the foreground but they are susceptible to disturbance such as camera rotation and plain image texture.

1) Motion Vector Calculation: When a new frame $\mathcal{F}^t$ is captured, existing block-matching motion estimation algorithms [18] can be used to calculate the set of motion vectors $\mathcal{M}^t$ for $\mathcal{F}^t$ using previous frame $\mathcal{F}^{t-1}$ as a reference frame.

2) Ego Motion Judgement: It is necessary to judge whether the mobile agent is in motion as Observation 1 stands only when the agent translates in space. To this end, we analyze the non-zero motion vector ratio of a frame, denoted as η, defined as the ratio of the number of macroblocks with non-zero motion vector to the number of total macroblocks. We randomly select 50 video clips from the nuScenes test set and manually divide video frames into two categories, *i.e.,* ego agent is moving and ego agent is static. We calculate η of each frame and perform a categorized analysis based on the motion state of ego agent. Figure 6(a) plots the cumulative distribution functions (CDFs) of η. It can be seen that, with



(a) CDFs of η for stopped ego vehicles and moving ego vehicles

(b) Motion estimation on an example driving video clip

Fig. 6: The non-zero motion vector ratio can be utilized for ego motion detection.

high probability (over 98%), whether the agent is in motion can be judged by simply comparing whether η is larger than a threshold (*e.g*, $\eta > 0.15$). Figure 6(b) depicts the η as a function of time in an example driving video clip where the ego agent comes to a stop for a while and then starts to accelerate. It can be seen that motion judged by η is highly consistent with the ground truth.

3) Rotational Component Elimination: If the mobile agent, such as a vehicle, can only translate along its z-axis (*i.e.*, $\Delta X = 0$ and $\Delta Y = 0$) and rotate around its x- and y-axis (*i.e.*, $\Delta\phi_z = 0$), motion vectors combining Eq.(3) and Eq.(5) can be simplified as

$$\begin{aligned} vx_q &= \frac{\Delta Z x_q^t}{Z_Q} - \Delta\phi_y f + \frac{\Delta\phi_x x_q^t y_q^t}{f} - \frac{\Delta\phi_y {x_q^t}^2}{f}, \\ vy_q &= \frac{\Delta Z y_q^t}{Z_Q} + \Delta\phi_x f - \frac{\Delta\phi_y x_q^t y_q^t}{f} + \frac{\Delta\phi_x {y_q^t}^2}{f}. \end{aligned} \tag{6}$$

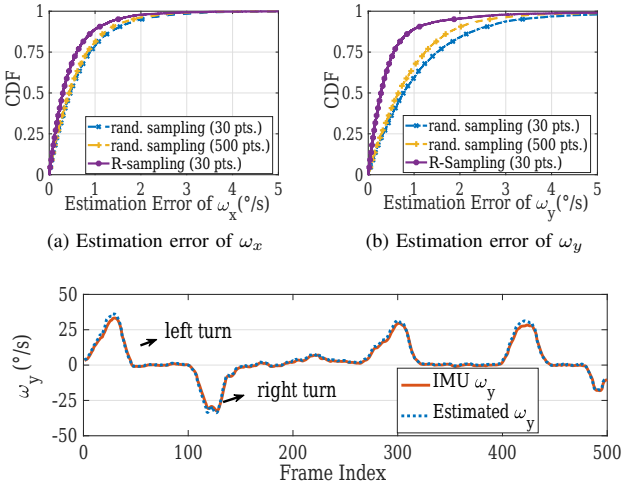
In Eq.(6), $\frac{\Delta Z}{Z}$ can be eliminated and we have

$$x_q^t f\Delta\phi_x + y_q^t f\Delta\phi_y = y_q^t vx_q - x_q^t vy_q, \tag{7}$$

where $\Delta\phi_x$ and $\Delta\phi_y$ can be obtained by solving over-determined linear equations with a set of more than two motion vectors.

As noisy motion vectors present, how to select such a set of motion vectors that can lead to accurate $\Delta\phi_x$ and $\Delta\phi_y$ is non-trivial. We propose a simple yet effective method, called R-sampling, where k motion vectors that have the shortest distance to the fixed FOE, calibrated when the agent moves forward, are selected. The rationale of R-sampling lies in the fact that these vectors have small translation components and are more sensitive to rotations. We then adopt RANSAC [19] to solve the over-determined linear equations.

We conduct experiments on KITTI to evaluate the performance of R-sampling. We randomly select 6 video clips, 6332 frames in the KITTI dataset to conduct our experiments and utilize IMU data to calculate the ground truth of rotation speed of ego agent at each frame. We set $k = 30$ in R-sampling and $k = 30, 500$ in random sampling, respectively. Figure 7(a) and (b) plot the CDFs of estimated errors of rotational speed around x- and y-axis of the camera, denoted as ω_x and ω_y, respectively. It can be seen from the results that R-sampling with 30 samples can achieve higher rotational



(a) Estimation error of ω_x

(b) Estimation error of ω_y

(c) Illustration of estimation of ω_y

Fig. 7: Efficiency illustration of R-sampling.

speed estimation accuracy, compared with random sampling with 500 points. Figure 7(c) also illustrates an instance of the estimated rotational speed ω_y.

Given the estimated $\Delta\phi_x$ and $\Delta\phi_y$, rotational components in Eq.(6) can be effectively removed.

C. Foreground Extraction

1) Ground Estimation: When motion vectors only contain translational component, we have one key observation as follows:

Observation 2. *Motion vectors of objects of the same height in the physical world have the same normalized magnitude.*

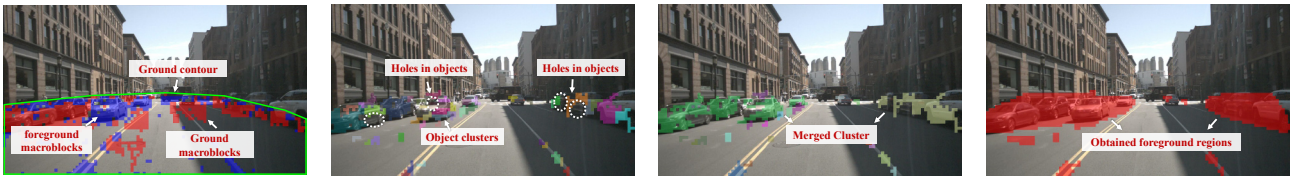
Let $R_{q^t} = \sqrt{(x_q^t - \frac{\Delta X f}{\Delta Z})^2 + (y_q^t - \frac{\Delta Y f}{\Delta Z})^2}$ denote the distance between an image point q on frame $\mathcal{F}^t$ and the FOE. Then, motion vector v_q^t can be normalized by $R_{q^t} y$,

$$\frac{v_q^t}{R_{q^t} y} = \frac{\Delta Z}{f Y_Q}. \tag{8}$$

It can be seen that the normalized vector is only related to Y_Q as ΔZ, f are the same for all points, which supports Observation 2.

As ground can be treated with the same and smallest height in the scene, to estimate ground, we need to identify these vectors with the smallest normalized magnitude. However, motion vectors in regions with plain texture can be rather noisy. Moreover, these motion vectors on the ground can hardly be absolutely equal. To tackle these issues, we first filter out those random vectors that do not point to the FOE and then adopt the Triangle method [20] to statistically establish a magnitude threshold. If a motion vector has a normalized magnitude that is less than the threshold, the corresponding macroblock is considered a ground macroblock.

After obtaining a set of ground macroblocks, we then utilize a convex hull generation algorithm, *e.g.*, Sklansky's algorithm [21] to generate the convex hull of the ground region. Non-ground macroblocks locate within the ground region in frame



(a) Estimated ground with foreground seed macroblocks (b) Region-growing-based object clustering with holes in identified objects (c) Similar foreground objects can be merged into one region (d) Final foreground obtained by taking convex hull of foreground regions

Fig. 8: Illustration of foreground estimation with corrected motion vectors.

$\mathcal{F}^t$ constitute a set of foreground seed macroblocks, denoted as $\mathcal{S}^t$. Figure 8(a) illustrates the result of ground estimation where red and blue cells represent ground and foreground macroblocks, and green polygon represents the ground convex.

2) Region-growing-based Foreground Clustering: Given $\mathcal{S}^t$, we utilize region-growing-based clustering algorithm to obtain foreground objects of interest. Specifically, for each macroblock $b_i \in \mathcal{S}^t$, a breadth first search (BFS) is conducted around the four neighboring macroblock of b_i. If the motion vector of a neighboring block b_j is similar to that of b_i and the average motion vector of the current cluster that b_i belongs to, b_j is appended to the cluster and the average motion vector of the current cluster is also updated. The latter condition is to avoid over-growing of an object. Figure 8(b) illustrates the object clustering result of the previous example, where different colors represent different foreground objects. As shown in Figure 8(b), many identified foreground objects may have holes as motion vectors are sparse and coarse. To obtain more complete foreground, as illustrated in Figure 8(c), we further iteratively merge clusters with similar directions of their average motion vectors until no more clusters can be merged. Finally, we use the Sklansky's algorithm to generate the convex contour for each merged cluster and obtain the foreground regions, as shown in Figure 8(d).

D. Adaptive Video Encoding

1) Uplink Bandwidth Estimation: The quality of wireless uplink varies as the mobile agent moves. To best encode the video, the agent needs to continuously estimate the uplink bandwidth over time. Specifically, the uplink bandwidth is estimated with the amount of encoded data that has been successfully transmitted to the edge server within a sliding time window of 2ms.

2) Optimal QP Assignment: Given the estimated foreground regions and the current uplink bandwidth, as introduced in Subsection II-B, the optimal QP assignment for each macroblock is equivalent to assign the optimal QP offset map to the foreground and the background macroblocks. We denote the optimal delta QP value between the foreground and the background macroblocks as δ. Based on the observation that larger extracted foregrounds are more likely to cover more real foregrounds, we design an adaptive δ which is proportional to the size of extracted foregrounds. More specifically, δ equals to current foreground size multiplying a constant coefficient. Then we fix the QP offset of foreground macroblocks to 0 and assign background macroblocks with a QP offset value equal to δ. After the frame $\mathcal{F}^t$ is encoded, it is put into the transmitting queue waiting to be sent to the edge server.

Name	# FPS	# videos	# frames	# cars	# peds.
nuScenes	12	50	9605	45605	10221
RobotCar	16	8	8150	19365	25423

TABLE I: Summary of datasets

E. Motion-vector-based Offline Tracking

When encountering link outages, such as hard handovers between base stations or multipath fading, the agent should be able to detect this situation and act accordingly. Specifically, a timer is set up when an encoded frame becomes the first frame in the transmitting queue. If the timer times up before the frame is sent out, the agent considers that a link outage occurs. In this case, local motion vector based tracking is conducted for this and after frames until the link is recovered. More specifically, the mean of those motion vectors that reside in a detected bounding box in the previous frame is calculated, which is used to move the detected bounding box from the previous location in frame $\mathcal{F}^{t-1}$ to the corresponding new location in frame $\mathcal{F}^t$.

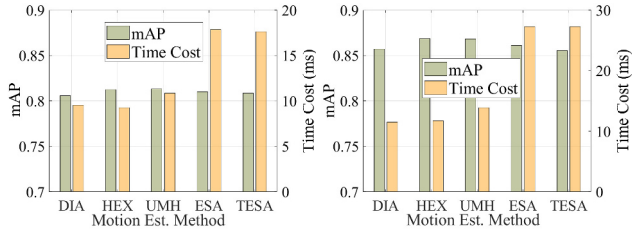
IV. Performance Evaluation

A. Methodology

We conduct our experiments on RobotCar and nuScenes to evaluate the performance of DiVE. For nuScenes, we remove video clips collected at night in test set as almost all motion vectors are calculated to be zero at night and randomly select 50 video clips from the rest video clips. For RobotCar, we randomly select 8 video clips, each consisting of approximately 1000 frames. The details of datasets are presented in Table I.

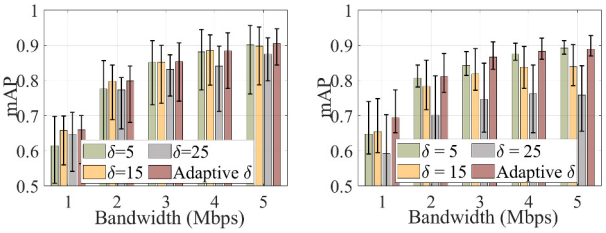
We compare DiVE with the following video analytics schemes:

- **O^3 [4]:** a mobile object detection system that uploads key frames to the cloud server for object detection and uses key frame detection results to correct object tracking results for other frames.
- **EAAR [3]:** an object detection system designed for mobile AR devices to conduct parallel streaming and inference on key frames. It utilizes the cached detection results of key frames for ROI encoding and conducts object tracking on other frames. The default QP is 30 and 40 for high-quality and low-quality encoding.



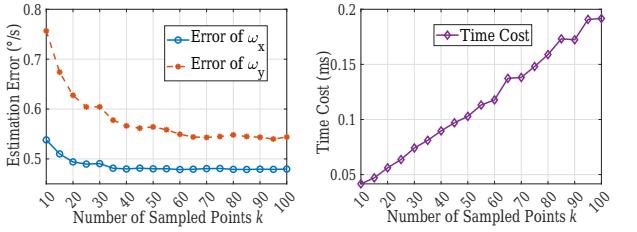
(a) mAP and time cost on RobotCar (b) mAP and time cost on nuScenes

Fig. 9: Effect of different motion estimation methods.



(a) mAP on RobotCar (b) mAP on nuScenes

Fig. 11: Effectiveness of Optimal QP Assignment.



(a) Estimation error with different number of sampled points k (b) Time cost with different number of sampled points k

Fig. 10: Effect of different number of sampled points k.

- **DDS [5]:** a video analytics system for deep learning inference, which first uploads low-quality videos to the cloud server for detection and utilizes the feedback detected regions to guide the edge device to re-upload the corresponding high-quality videos of the corresponding part to the cloud server for more accurate detection.

For fair comparison, we utilize *x264* for video/key frame encoding in all schemes. For O^3 and EAAR, we utilize the same motion vector based tracking methods. For DDS, we use the frame-level transmission which is the same as DiVE instead of raw segment-level transmission which will lead to high latency.

We consider the following two metrics to evaluate the performance of different schemes:

- **Precision:** we utilize Average Precision (AP), a popular metric for the object detection task, to represent the precision of DNN inference results. We utilize the object detection results of raw frames at the edge server as the ground truth to calculate AP.
- **Response Time:** we define Response Time as the average duration from the moment a frame is captured by the camera until the mobile agent receives the final inference result.

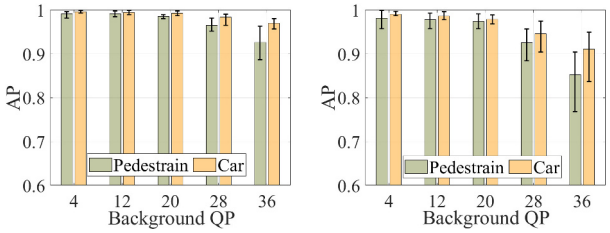
B. Parameter Configuration

1) Effect of Motion Estimation Method: We execute a set of experiments to evaluate the performance of different motion estimation methods. In the *x264* encoder, there are five motion estimation methods, *i.e.,* diamond search algorithm (DIA), hexagon-based search algorithm (HEX), uneven multi-hexagon search algorithm (UMH), transformed exhaustive search algorithm (TESA) and exhaustive search algorithm (ESA), ranked in ascending order of computational complexity. We set the bandwidth to 2Mbps and use each motion estimation method to extract motion vectors when encoding each frame on RobotCar and nuScenes. Figure 9 plots the mAP and time cost of each motion estimation method on two datasets. The result shows that HEX and UMH achieve almost the same mAP and higher than that of DIA, ESA and TESA while HEX has lower time cost than UMH on both datasets. The reason is that simple motion estimation method like DIA will lead to low quality of motion vectors while overcomplicated motion estimation methods like ESA and TESA will also lead to more noise in motion vectors as motion estimation methods are designed for obtaining minimal residual data but not real object matching. Therefore, we choose HEX for motion estimation in DiVE to achieve a good trade-off between time cost and quality of calculated motion vectors.

2) Effect of the Number of Sampled points in R-sampling: In this experiment, we explore the effect of number of R-sampling points k in Rotational Component Elimination. We conduct rotation estimation on KITTI and vary the number of sampled points k from 10 to 100 with an interval of 5. We utilize the same video clips as experiments in III-B. We calculate the estimation error of rotational speed and time cost of RANSAC under each setting. Figure 10 plots the results. It can be seen from the result that the estimation error decreases when k increases and the time cost is linear to k. Moreover, the estimation error converges when k reaches 70. The reason is that R-sampling is efficient and a few accurate motion vectors will lead to high motion estimation accuracy, more coarse motion vectors will not increase the motion estimation accuracy any more. As a result, we choose $k = 70$ in R-sampling to achieve good rotation estimation accuracy while maintaining as low time cost as possible.

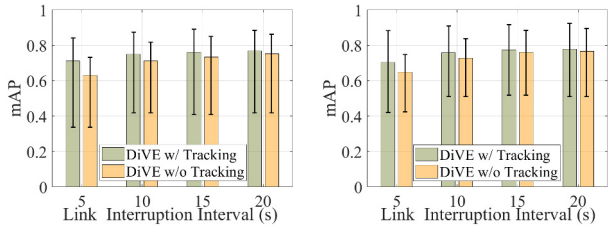
C. Effectiveness of Optimal QP Assignment

We conduct experiments on RobotCar and nuScenes to evaluate the effectiveness of Optimal QP Assignment. We vary δ from 5 to 25 with an interval of 10 along with adaptive δ in our design and bandwidth from 1Mbps to 5Mbps with an



(a) AP on RobotCar (b) AP on nuScenes

Fig. 12: Effectiveness of Foreground Extraction.



(a) mAP on RobotCar (b) mAP on nuScenes

Fig. 13: Effectiveness of Motion-vector-based Offline Tracking.

interval of 1Mbps. Figure 11 plots the mAP with different δ under each bandwidth on both two datasets. It can be seen from the result that adaptive δ achieves the highest mAP under most bandwidth settings. Moreover, adaptive δ achieves much better performance than $\delta = 5$ when the bandwidth is 1Mbps and the reason is that higher bitrate for foregrounds can lead to more inference accuracy benefits when the bandwidth is low. The experiment results demonstrate the efficiency of Optimal QP Assignment.

D. Effectiveness of Foreground Extraction

In this experiment, we set *x264* encoder to Constant Rate Factor (CRF) mode which can fix the macroblock encoding quality with fixed QP value and fix the QP value of foregrounds to 0 to achieve almost the same quality with raw frames. The QP value of background macroblocks varies from 4 to 36 with an interval of 8. Figure 12 plots the AP of varying QP value of backgrounds on RobotCar and nuScenes. The result shows that the AP decreases slowly with the increase of the QP value of backgrounds. When the QP value increases to 20, the AP of the pedestrian and car is 0.985, 0.993 on RobotCar, and 0.974, 0.979 on nuScenes, which shows nearly no loss on the accuracy of detection. Even when the QP value of backgrounds increases to 36 which leads to severe quality loss of video frames, the AP of pedestrian and car can still be up to 0.926, 0.970 on RobotCar, and 0.852, 0.911 on nuScenes. The result demonstrates the effectiveness of Foreground Extraction in DiVE.

E. Effectiveness of Motion-vector-based Offline Tracking

We conduct experiments on RobotCar and nuScenes to evaluate the effectiveness of Motion-vector-based Offline Tracking in DiVE. We set the bandwidth to 2Mbps and add 1s link interruptions, *i.e.,* bandwidth drops to 0Mbps to the network. The interval between two interruptions is set from 5s to 20s with an interval of 5s to simulate different mobile wireless network scenarios. Figure 13 plots the mAP of detection results with and without offline tracking in different network scenarios on two datasets. It can be seen from the results that when offline tracking is enabled, the mAP increases on all network scenarios. Specifically, the mAP increases by 12.8% and 8.6% in detection accuracy on RobotCar and nuScenes with link interruption interval of 5s when Motion-vector-based Offline Tracking is enabled. The result shows that when Motion-vector-based Offline Tracking is enabled, DiVE can still maintain high inference accuracy when wireless link interrupts in a short time, while accuracy decrease occurs without offline tracking, which demonstrates the effectiveness of Motion-vector-based Offline Tracking.

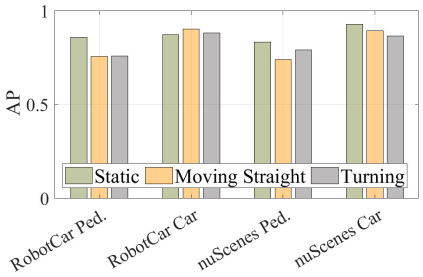
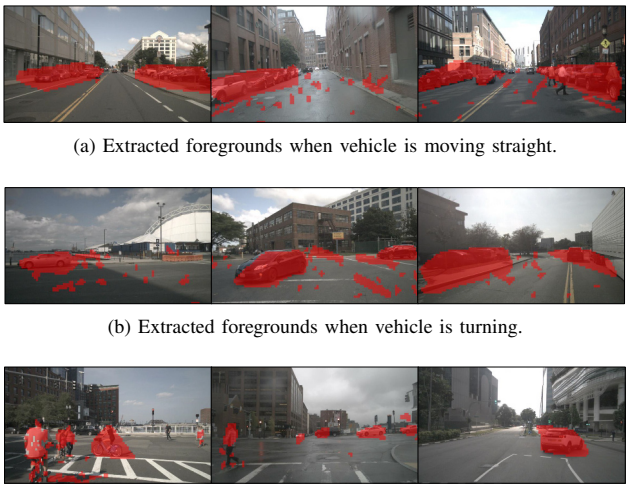


Fig. 14: Impact of different motion states.



(a) Extracted foregrounds when vehicle is moving straight.

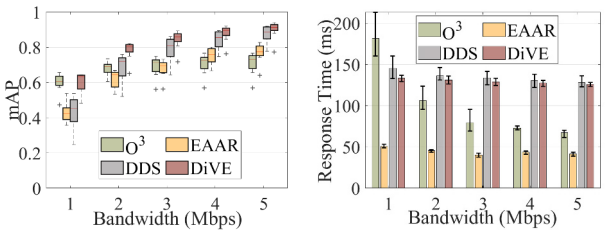
(b) Extracted foregrounds when vehicle is turning.

(c) Extracted foregrounds when vehicle is static.

Fig. 15: Illustration of extracted foreground under different motion states.

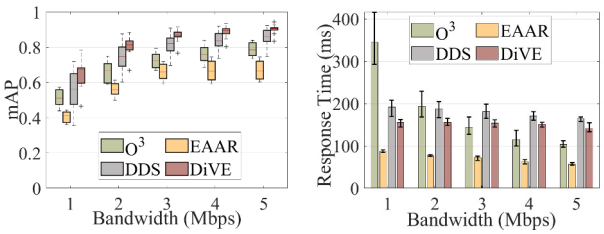
F. Impact of Different Motion State

We manually divide RobotCar and nuScenes into three different motion states, *i.e.,* static, moving straight, turning, and record the AP of pedestrian and car detection results under different motion states in Figure 14. In these experiments,



(a) mAP on RobotCar (b) Response Time on RobotCar

Fig. 16: Comparison of different schemes under different bandwidth on RobotCar.



(a) mAP on nuScenes (b) Response Time on nuScenes

Fig. 17: Comparison of different schemes under different bandwidth on nuScenes.

video bitrate is set to 2Mbps. The result shows that DiVE consistently achieves pedestrian detection AP exceeding 0.6 across all datasets, with an average pedestrian detection AP of 0.847, 0.750 and 0.777 when the ego vehicle is stationary, moving straight and turning. The car detection AP is higher than 0.8 in all datasets and the average car detection AP is 0.901, 0.898 and 0.874 when the ego vehicle is static, moving straight and turning. Moreover, the highest car detection AP is achieved when the ego vehicle is static. The reason is that the ego vehicle stops in most cases due to a red light and other moving vehicles can be captured as foreground more accurately. Figure 15 illustrates some samples of extracted foregrounds when the ego vehicle is moving straight, turning and static.

G. End-to-end Performance Evaluation

We evaluate the end-to-end performance of different video analytics schemes on accuracy and response time across varying network scenarios on RobotCar and nuScenes. Figure 16 and Figure 17 plot the AP and response time of different schemes across varying netwok bandwidth from 1Mbps to 5Mbps with an interval of 1Mbps. It can be seen from the result that DiVE achieves the highest mAP under each network scenario on both RobotCar and nuScenes. When compared to DDS, which exhibits the most similar performance with DiVE, DiVE improves mAP by 2.8%-39.1% on RobotCar and by 4.7%-17.6% on nuScenes. Notably, DiVE outperforms other schemes especially when bandwidth is low.

Figure 16(b) and Figure 17(b) show that DiVE can achieve lower than 134ms/156ms response time under different bandwidth, which is 1.7%-8.4% and 14.0%-19.1% lower than DDS on RobotCar and nuScenes, respectively. Although EAAR can achieve lower response time than DiVE, its detection mAP is significantly lower than DiVE. We explain the above results as follows: O^3 and EAAR achieve poor precision under different bandwidth because they do not utilize the temporary redundancy in the video. Instead, DiVE captures this redundant information and utilizes them to aid in differential video encoding. Although local tracking has a very low time cost, prolonged tracking will lead to a severe accuracy decrease. The DDS scheme requires two times of uploading for final detection results, which leads to high response time. The end-to-end experiment results demonstrate the superiority of DiVE compared with other schemes.

V. Related Work

Traditional Video Streaming: Plenty of research is done to increase the Quality of Experience (QoE) of users for watching video under fluctuated bandwidth [22]–[27]. Sun *et al.* [22] proposed a bandwidth prediction system called CS2P which clusters current network bandwidth to a pre-learned cluster and utilizes a HMM model to predict the evolution of bandwidth for proper bitrate selection of video. Mao *et al.* [23] proposed a reinforcement learning based video streaming system called Pensieve which utilizes a neural network to select bitrates for future video chunks based on observations of history bandwidth. Yeo *et al.* [24] proposed a video streaming system called NAS which utilizes a super-resolution model at client side to up-scale received low-resolution video frames for better QoE. Yeo *et al.* [26] designed a system called NEMO which applies neural super-resolution to a few selected frames and utilizes the outputs to benefit other frames. These video streaming schemes are designed to improve QoE of humans under various bandwidth and do not consider about how to keep high accuracy of DNN inference on mobile agents.

Video Analytics Systems: Differing from traditional video streaming system designed for humans. A large number of studies focus on maintaining high accuracy of DNN inference at edge device by uploading video to the cloud server for inference [2], [4]–[12], [28]–[32]. Some studies try to reconfigure video encoding parameters like resolution, frame rate and QP value during streaming to meet the bandwidth variation. Zhang *et al.* [11] designed a system called AWStream which combines offline and online profiling to choose encoding parameters. Li *et al.* [12] designed Reducto, an on-camera filtering system which filters out frames based on per frame difference. However, such schemes do not consider the importance difference inner frame and usually need offline profiling which can not be utilized for real-time video analytics.

Another direction is to choose key frames for uploading and inference based on difference between frames and conduct optical flow/motion vector based tracking with cached key frame detection results locally [2]–[4]. Such methods do not fully utilize the temporary redundancy between video frames and simple object tracking will lead to severe accuracy degradation on long frame sequences.

Besides, running small models on edge device for assistance like large object detection [10], estimation of importance of macroblocks [9] is also proposed in previous studies. Such

methods require for high computility at edge device and can not adapt to various mobile scenarios due to poor performance of cheap model.

Some studies also focus on foreground extraction and utilize differential encoding to reduce bandwidth consumption. Wang *et al.* [32] designed a system called VaBUS which constructs the background region of the surveillance camera and uploads differentially encoded frames. Wang *et al.* [31] proposed an algorithm called Orchestra, which divides video into different zones and assigns different qualities for encoding. However, these methods can only work with static backgrounds like video captured by surveillance camera and can not copy with highly dynamic mobile scenarios.

VI. Conclusion

In this paper, a video analytics system designed for mobile agents called DiVE is proposed. In DiVE, mobile agent determines its motion state and extracts interested foregrounds based on low-cost motion vectors calculated by video codec for differential encoding. Multiple observations and results of empirical studies are utilized to correct motion vectors and cluster foregrounds. As a result, DiVE can copy with coarse motion vectors and extract foregrounds accurately. Extensive experiment results demonstrate that DiVE can achieve high inference accuracy with low response time under various network scenarios.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China (Grants No. 62432008, 62472083) and Natural Science Foundation of Shanghai (Grant No. 22ZR1400200).

References

[1] X. Wang, J. Ye, and J. C. Lui, "Online learning aided decentralized multi-user task offloading for mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 4, pp. 3328–3342, 2024.

[2] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *Proceedings of ACM Sensys*, pp. 155–168, 2015.

[3] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *Proceedings of ACM SIGCOMM*, pp. 1–16, 2019.

[4] M. Hanyao, Y. Jin, Z. Qian, S. Zhang, and S. Lu, "Edge-assisted online on-device object detection for real-time video analytics," in *Proceedings of IEEE INFOCOM*, pp. 1–10, 2021.

[5] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang, "Server-driven video streaming for deep learning inference," in *Proceedings of ACM SIGCOMM*, pp. 557–570, 2020.

[6] 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 *Proceedings of ACM Mobicom*, pp. 201–214, 2021.

[7] S. Liu, T. Wang, J. Li, D. Sun, M. Srivastava, and T. Abdelzaher, "Adamask: Enabling machine-centric video streaming with adaptive frame masking for dnn inference offloading," in *Proceedings of ACM MM*, pp. 3035–3044, 2022.

[8] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Noscope: Optimizing neural network queries over video at scale," *VLDB Endowment*, vol. 10, no. 11, 2017.

[9] K. Du, Q. Zhang, A. Arapin, H. Wang, Z. Xia, and J. Jiang, "Accmpeg: Optimizing video encoding for accurate video analytics," in *Proceedings of Machine Learning and Systems*, pp. 450–466, 2022.

[10] Z. Yang, X. Wang, J. Wu, Y. Zhao, Q. Ma, X. Miao, L. Zhang, and Z. Zhou, "Edgeduet: Tiling small object detection for edge assisted autonomous mobile vision," *IEEE/ACM Transactions on Networking*, pp. 1–14, 2022.

[11] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "Awstream: Adaptive wide-area streaming analytics," in *Proceedings of ACM SIGCOMM*, pp. 236–252, 2018.

[12] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-camera filtering for resource-efficient real-time video analytics," in *Proceedings of ACM SIGCOMM*, pp. 359–376, 2020.

[13] Y. Cheng, Z. Zhang, H. Li, A. Arapin, Y. Zhang, Q. Zhang, Y. Liu, K. Du, X. Zhang, F. Y. Yan, *et al.*, "Grace: Loss-resilient real-time video through neural codecs," in *Proceedings of USENIX NSDI*, pp. 509–531, 2024.

[14] L.-L. Wang, W.-H. Tsai, *et al.*, "Camera calibration by vanishing lines for 3-d computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 370–376, 1991.

[15] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*, pp. 3354–3361, IEEE, 2012.

[16] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of IEEE CVPR*, pp. 11621–11631, 2020.

[17] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.

[18] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287–290, 2000.

[19] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[20] G. W. Zack, W. E. Rogers, and S. A. Latt, "Automatic measurement of sister chromatid exchange frequency.," *Journal of Histochemistry & Cytochemistry*, vol. 25, no. 7, pp. 741–753, 1977.

[21] J. Sklansky, "Finding the convex hull of a simple polygon," *Journal of Algorithms*, vol. 1, no. 2, pp. 79–83, 1982.

[22] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of ACM SIGCOMM*, pp. 272–285, 2016.

[23] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of ACM SIGCOMM*, pp. 197–210, 2017.

[24] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han, "Neural adaptive content-aware internet video delivery," in *Proceedings of USENIX OSDI*, pp. 645–661, 2018.

[25] X. Zhang, Y. Ou, S. Sen, and J. Jiang, "Sensei: Aligning video streaming quality with dynamic user sensitivity," in *Proceedings of USENIX NSDI 21*, pp. 303–320, 2021.

[26] H. Yeo, C. J. Chong, Y. Jung, J. Ye, and D. Han, "Nemo: Enabling neural-enhanced video streaming on commodity mobile devices," in *Proceedings of ACM Mobicom*, 2020.

[27] J. Kim, Y. Jung, H. Yeo, J. Ye, and D. Han, "Neural-enhanced live streaming: Improving live video ingest via online learning," in *Proceedings of ACM SIGCOMM*, pp. 107–125, 2020.

[28] T. Zhang, A. Chowdhery, P. V. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," in *Proceedings of ACM Mobicom*, pp. 426–438, 2015.

[29] T. Yuan, L. Mi, W. Wang, H. Dai, and X. Fu, "Accdecoder: Accelerated decoding for neural-enhanced video analytics," in *Proceedings of IEEE INFOCOM*, pp. 1–10, 2023.

[30] Y. Wang, W. Wang, J. Zhang, J. Jiang, and K. Chen, "Bridging the Edge-Cloud barrier for real-time advanced vision analytics," in *Proceedings of USENIX HotCloud Workshop*, 2019.

[31] W. Wang, B. Wang, L. Zhang, and H. Huang, "Sensitivity-aware spatial quality adaptation for live video analytics," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 8, pp. 2474–2484, 2022.

[32] H. Wang, Q. Li, H. Sun, Z. Chen, Y. Hao, J. Peng, Z. Yuan, J. Fu, and Y. Jiang, "Vabus: Edge-cloud real-time video analytics via background understanding and subtraction," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 1, pp. 90–106, 2023.