CLOUD-CODEC: A New Way of Storing Traffic Cameras Footage at Scale

HOYOUNG KIM*, Stony Brook University, USA
AZIMBEK KHUDOYBERDIEV*, The State University of New York, Korea and Stony Brook University, USA
SHUBHANGI S. R. GARNAIK, The State University of New York, Korea and Stony Brook University, USA
ARANI BHATTACHARYA, Indraprastha Institute of Information Technology Delhi, India
JIHOON RYOO†, The State University of New York, Korea

Storing large volumes of traffic video content in cloud storage is an expensive undertaking, given the limited capacity of cloud storage and its inability to store data beyond a few weeks. To address this issue, this paper introduces *CLOUD-CODEC*, a novel video encoding approach tailored specifically for traffic monitoring video. *CLOUD-CODEC* offers three key advantages: (i) real-time encoding without any delay, (ii) near-perfect video quality upon decoding, and (iii) one-fifth the storage size of traditional encoding methods. *CLOUD-CODEC* is generally applicable to traffic cameras under various weather and lighting conditions. The encoding algorithm is a lightweight DNN-based object detection and box shaped segmentation approach. The method can uniquely detect and segment cars, pedestrians, and moving objects with the marginal box shaped contours. Periodic object detection makes it possible for *CLOUD-CODEC* to operate in real-time and estimate the movement of objects between predictions. Proof-of-concept evaluations using a massive dataset indicate that *CLOUD-CODEC* reduces video size by 80%—surpassing AV1 (34.9%), CloudSeg (58.4%), Detection (76.9%), Segmentation (73.1%), and Segm&Sort (69.5%). It achieves a frame rate of 95.8 when encoding, and a VMAF score of 72.54 after decoding, with a storage size that is one-fifth of traditional methods. Field-testing of *CLOUD-CODEC* on metropolitan traffic cameras demonstrates its ability to extend storage time by 74.92 percent.

CCS Concepts: • **Information systems → Storage management**; *Multimedia information systems*; • **Computing methodologies → Computer vision problems**.

Additional Key Words and Phrases: Traffic Video Storage, Video Encoding, Foreground Extraction, Background Extraction, Cloud Storage Optimization, Dynamic Object Detection

ACM Reference Format:
Hoyoung Kim, Azimbek Khudoyberdiev, Shubhangi S. R. Garnaik, Arani Bhattacharya, and Jihoon Ryoo. 2025. *CLOUD-CODEC*: A New Way of Storing Traffic Cameras Footage at Scale. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 28 pages. https://doi.org/XXXXXXX.XXXXXXX

*Both authors contributed equally to this research.
†Corresponding author: Jihoon Ryoo

Authors' Contact Information: Hoyoung Kim, hoyokim@cs.stonybrook.edu, Stony Brook University, Stony Brook, New York, USA; Azimbek Khudoyberdiev, azimbek.khudoyberdiev@stonybrook.edu, The State University of New York, Incheon, Korea and Stony Brook University, Stony Brook, New York, USA; Shubhangi S. R. Garnaik, shubhangisaile.garnaik@stonybrook.edu, The State University of New York, Incheon, Korea and Stony Brook University, Stony Brook, New York, USA; Arani Bhattacharya, arani@iiitd.ac.in, Indraprastha Institute of Information Technology Delhi, Delhi, India; Jihoon Ryoo, jihoon.ryoo@sunykorea.ac.kr, The State University of New York, Incheon, Korea.

Manuscript submitted to ACM

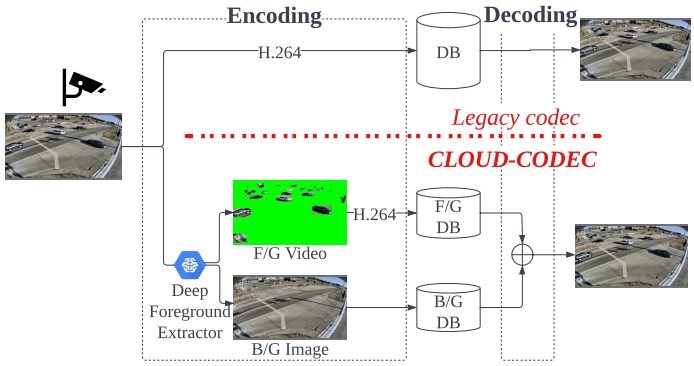


Fig. 1. System Flow of *CLOUD-CODEC*.

1 INTRODUCTION

As urban populations continue to grow and cities become increasingly interconnected, the need for efficient and effective traffic management has never been more crucial. Traffic surveillance systems employing CCTV cameras have emerged as indispensable tools for monitoring, analyzing, and managing urban traffic flow [32, 35]. These systems not only contribute to improved public safety and security but also facilitate data-driven decision-making for urban planning and transportation policies. Moreover, they are essential for monitoring traffic conditions, detecting accidents, and enforcing traffic laws, which can ultimately lead to a reduction in congestion and improve overall transportation efficiency [24, 34, 76].

However, the proliferation of high-definition traffic surveillance cameras has generated an immense volume of video data, which poses significant challenges in terms of storage and bandwidth requirements [23, 59]. Conventional video encoding methods, while widely used, often struggle to strike the right balance between storage efficiency and video quality, particularly in real-time traffic monitoring scenarios. The limitations of these methods have a direct impact on the practicality and effectiveness of traffic surveillance systems.

Furthermore, the rapid growth in video data outpaces the limited capacity of cloud storage, which typically retains traffic videos for only a few weeks. This short retention period may hinder the investigation of older incidents and limit the utility of traffic data for long-term analysis and planning. In addition, the cost of storing large volumes of traffic video content in cloud storage can be quite substantial, making it financially impractical for many municipalities and organizations to maintain extensive archives of traffic videos [2, 56, 62].

To address these challenges, this study introduces a novel video encoding approach, *CLOUD-CODEC*, specifically tailored for traffic monitoring video. *CLOUD-CODEC* leverages a lightweight deep neural network (DNN)-based object detection and box shaped segmentation algorithm to achieve real-time encoding, near-perfect video quality upon decoding, and a significant reduction in storage size compared to traditional encoding methods. By uniquely detecting and segmenting cars, pedestrians, and other moving objects with box shaped marginal contours, *CLOUD-CODEC* can accurately estimate object movement between real-time predictions.

CLOUD-CODEC offers significant industrial advantages in managing large-scale traffic surveillance footage through its innovative background-foreground extraction techniques. Key benefits include a reduction in data volume by encoding dynamic moving objects and static background separately, which alleviates storage demands and allows organizations to manage extensive surveillance data efficiently. This approach leads to cost optimization for long-term storage, making it a financially viable solution for city governments and private entities by lowering expenses related to data centers and

cloud services. Urban areas, especially expanding cities, benefit from *CLOUD-CODEC* ability to retain video footage for extended periods to meet legal, security, and operational needs without a substantial increase in storage costs, supporting the preservation of historical footage over time. The proposed approach has been extensively evaluated using a massive dataset, demonstrating impressive performance in terms of encoding frame rate, video quality, and storage size reduction. Moreover, *CLOUD-CODEC* has been tested under various weather and lighting conditions, showcasing its adaptability and robustness in real-world traffic surveillance applications. Field-testing of *CLOUD-CODEC* on metropolitan traffic cameras further highlights its potential to extend storage time by a substantial percentage, thereby addressing the critical challenge of limited storage capacity in traffic monitoring systems.

We summarize our contributions as follows:

- We identify that isolating and extracting dynamic foreground objects from largely redundant static background content enables substantial video size reduction without losing critical information. To the best of our knowledge, this is the first work to apply a foreground-background extraction approach to reduce storage size in traffic surveillance.
- We introduce a Deep Foreground Extractor that efficiently identifies moving objects, separating video into dynamic foreground elements and static background scenes, which are stored independently to reduce storage requirements (Fig. 1).
- We examine four different strategies for Deep Foreground Extractor; object detection, instance segmentation, box interpolation and trajectory prediction. We adopt box interpolation approach based on two aspects, time-efficiency and storage saving.

The remainder of the paper is structured as follows. Section 2 presents the background and a literature review of encoding solutions. Section 3 details the methodology and key components of the DFE algorithms, Section 4 describes the implementation and datasets used. Evaluation results based on various performance metrics are detailed in Section 5. Finally, the remaining sections discuss the limitations, future research directions, and conclusion of the paper.

2 BACKGROUND AND RELATED WORKS

2.1 Background

Several video-storing solutions have been developed and introduced in the past decades to optimize video-storing processes. Video compression (content and block) [8, 10, 28, 52] and motion segmentation [57, 67] techniques are two of the most popular and active topics in this area. The video compression technique minimizes the amount of storage required for recorded footage and reduces the transmission bandwidth over a network. In the block-based video compression technique, the video frame is divided into small, fixed-sized blocks, and the data is compressed within each block. As a result, the redundancy and static patterns in the video content are easily identified and eliminated. The content-based video compression, on the other hand, analyzes the entire frame and identifies regions that accommodate predictable or redundant information [18]. `H.264/AVC`, `H.265/HEVC`, `VP9`, and `AV1` are examples of the most common video compression formats [50]. `BCBR` proposed a block-based learning mechanism for high-efficient video coding by introducing Block-Composed Background Reference (BCBR) [13].

Motion segmentation [33, 54] is also one of the most well-known technologies used in video size reduction by dividing video content into regions and objects that are in motion independently. By analyzing the moving objects, the redundancy, and predictability of the video content provide remarkable efficiency for compression algorithms. Optical flow [15], background subtraction [3], object tracking [39], and clustering [43] methods are the most active research areas in motion

segmentation-based video size compression applications due to the requirement for transmission and storage of enormous amounts of data. However, existing motion segmentation struggles with inaccurate estimations when objects in the video move quickly or in complex ways. Furthermore, motion segmentation heavily relies on inter-frame prediction, meaning that a single frame's quality can depend on the quality of previous frames. This leads to the degradation of video quality over the period, especially if there are errors or dropped frames in video streaming.

To improve video storage efficiency further, shot boundary detection (SBD) and video summarization have been explored. Traditional SBD methods face high computational costs, limiting real-time use. To address this, recent work [31] introduced a fast SBD method leveraging separable moments and support vector machines (SVM), significantly reducing computation time while maintaining high detection accuracy. Meanwhile, deep learning-based video summarization [55] offers an alternative for reducing storage while preserving key content. These advancements complement existing compression and segmentation techniques, offering promising directions for optimizing large-scale video storing solutions.

2.2 Related Works

Existing video storage strategies fall into two categories – better encoding strategies and filtering of content on the camera side.

Encoding Strategies: Ma *et al.* proposed a library-based video coding scheme to enhance compression for traffic surveillance videos by leveraging content redundancy, using offline and online stages to extract, store, and match reference images for encoding, though it faces challenges with library complexity and environmental variability [19]. Deep learning approaches like Residual Squeeze-and-Excitation Network (RSE-Net) improve video quality through non-linear mapping with minimal parameters [19], while Du *et al.* introduced a joint compression and recognition model using Gaussian Mixture Models for structured storage and content analysis [20]. Yeo *et al.* developed `NeuroScaler`, a neural network framework for live stream quality enhancement by reducing super-resolution and encoding overheads [74], while `AccMPEG` focuses on improving quality in regions of interest [22]. The most related work, MPEG-4's `sprite coding` [16, 58], encodes static backgrounds as panoramic sprites separate from moving objects but struggles with global motion estimation, frame stitching, and dynamic scene changes. Unlike these methods, *CLOUD-CODEC* uniquely learns object presence to encode videos more efficiently.

Filtering on Camera-side: On-camera filtering approaches for video compression have been gaining remarkable attention to solve typical challenges in bandwidth, storage, and computation resources. Due to filtering out redundant or irrelevant frames at the camera level, the amount of data that needs to be sent, stored, and processed on the edge or back-end servers can be significantly reduced. `Reducto`, a resource-efficient on-camera filtering system, was introduced to dynamically adapt filtering decisions based on time-varying correlations among query accuracy, filtering threshold, low-level video feature type, and video content various parameters of real-time video [36]. `Reducto` achieved up to $51-97\%$ frame reduction on the camera side, which indeed allowed remarkable bandwidth and storage consumption. Kondi *et. al* designed `SmartFilter`, a real-time application-guided edge system for filtering video frames [60]. These approaches discard frames that are redundant on the camera-side while sending the rest to the main server for further analysis.

Server-driven streaming (`DDS`) was proposed to provide feedback on the quality of videos needed, and accordingly adjust the camera parameters [21]. The `DDS` continuously forwards low-quality video streams to the server, and the DNN on the server is used to determine which video chunks must be re-forwarded for better quality and higher inference accuracy. `CloudSeg` sends low-resolution frames and then applies super-resolution to obtain high recover high-resolution frames [66]. Lu *et al* introduced learning-based optical flow estimation for an end-to-end video compression framework [42]. The deep reinforcement learning-based approach, `CASVA` was proposed to make video configuration

choices by learning the relationship between inference accuracy and bandwidth requirement [75]. `CrossRoI` introduces region of interest optimization to remove the redundant appearance of same objects in multiple cameras by establishing offline cross-camera correlations and online real-time video inference [27]. `TILECLIPPER` introduced a system that employs tile sampling—transmitting only specific rectangular frame areas, or *tiles*, to the server [12]. `TILECLIPPER` adaptively selects these tiles based on their correlation with tile bitrates.

Most studies in the related literature primarily focus on reducing bandwidth consumption, which indirectly reduces storage requirements. While these methods achieve short-term storage savings, their effectiveness is limited for large-scale deployments, such as extensive traffic surveillance networks, where the cumulative volume of data remains substantial. In contrast, *CLOUD-CODEC* is specifically designed to efficiently encode large volumes of traffic video data at a city-wide scale and increase the retention period.

3 *CLOUD-CODEC*

In general, CCTV videos are collected on the database and require a massive amount of storage, and videos are deleted after short retention time. Therefore we propose a novel approach to save more videos and longer retention time by extracting dynamic moving objects (vehicles, pedestrians) from the static background. The static background remains the same in the video thus it is captured and saved periodically to reflect slowly switching light and environment changes. Whereas, the dynamic foreground moving objects are extracted using deep learning methods and saved to the database at full frame speed.

Figure 1 provides a brief overview of the overall system flow of *CLOUD-CODEC*. *CLOUD-CODEC* is used to dramatically reduce storage space. To ensure efficiency in processing numerous video streams and to consider quality during video recovery, *CLOUD-CODEC* offers four foreground extraction methods explained in Section 3.1. The background image is created by calculating the median from samples of video frames. When restoring a video, one can achieve results that closely resemble the original by simply blending the foreground and background together.

3.1 Foreground Extraction

To reduce video size with background removal, it is important to remove noise. Noise is typically affected by weather conditions, movement of trees or waters, camera shake, and night lighting. There are various studies on the subject of background subtraction to address this challenge, and recently, deep learning has been used to improve its quality. However, most models have slow processing speeds and it's difficult to find a model that produces consistent quality in various environments. Therefore, we implemented the Deep Foreground Extractor (DFE) to efficiently perform foreground and background extraction based on a state-of-the-art You Only Look Once (YOLO) architecture [64].

The proposed DFE supports both an instance segmentation model and an object detection model, and it performs at a speed close to real-time. Object detection localizes the object's position using bounding boxes. Instance segmentation accurately segments the area of the detected object, not just the bounding box.

We implemented four versions of the deep foreground extractor in *CLOUD-CODEC* based on (i) object detection, (ii) object detection and box interpolation, (iii) instance segmentation, and (iv) instance segmentation and trajectory prediction. Based on performance comparison, we decided which architecture *CLOUD-CODEC* uses.

To summarize, we have adopted the model that combines object detection and box interpolation in our system. As we will discuss further in Section 5, it has shown relatively superior performance in terms of inference speed and file size reduction.

(a) Bounding Box. (b) Silhouette. (c) Silhouette with Morphological Dilation.

Fig. 2. Three Types of Foreground Extraction.

Object Detection. In this approach, we selectively retain the regions corresponding to detected objects within each video frame, removing the RGB information outside the bounding boxes generated by the object detector, as shown in Figure 2(a). For each frame, a YOLO-based model detects objects, generating bounding boxes around them. A binary mask is then applied to the frame, zeroing out all pixels outside these bounding boxes and replacing them with a green background. This modified frame highlights only the detected objects, effectively discarding static background information and preserving only the foreground activity for storage. Even though this approach reduces data storage by retaining only detected objects, but it has notable drawback. Its frame-by-frame processing increases computational cost and lowering FPS, which limits its suitability for real-time applications or high-volume video data. Additionally, artifacts like partial shadows or nearby objects within bounding boxes can reduce the precision of the extracted foreground. Algorithm 4 in Appendix (§A) provides a detailed description of this object detection-based foreground extraction method.

Object Detection and Box Interpolation To build a faster and lighter system, we introduce box interpolation. The object detection is already real-time enough with an average 27.65 FPS, but the computational cost is high to process all the video streams from numerous cameras installed in urban areas. Box interpolation is used to solve this problem. It estimates the object's box as the midpoint of the size and position of the bounding boxes from the previous and next frames. When an object i's bounding box on frame t is $\mathbf{B}_t^i$,

$$\mathbf{B}_t^i = \frac{\mathbf{B}_{t+1}^i + \mathbf{B}_{t-1}^i}{2}, \quad where\ \mathbf{B}_t^i = [x_t^i\ y_t^i\ w_t^i\ h_t^i]. \tag{1}$$

$\mathbf{B}_t^i$ includes position x_t^i and y_t^i and size width w_t^i and height h_t^i of the bounding box. As shown in Figure 3(b), this reduces the frequency of model inference by 1/2, ensuring fast processing speed.

The pseudo-code for this approach is presented in Algorithm 1. For each input video, we apply object detection at every other frame, generating bounding boxes for detected objects. The SORT tracker then associates these detections across frames, ensuring consistent tracking of objects over time. Between detection frames, we interpolate bounding boxes to smooth the object trajectories and reduce flickering effects. A binary mask is created from these bounding boxes and applied to each frame, removing pixel information outside the detected regions. Non-detected areas are replaced with a green background.

Instance Segmentation Similar to object detection, we delete the RGB information of the portion outside the object's silhouette. Since the contour of the object is precisely traced, shadows or unnecessary information do not appear in the frame, presented in Figure 2(b). As a result, we can expect a higher compression rate than simple object detection. However, the downside is that the inference speed is slower compared to the object detection model, due to processing all of the frames as visualized in Figure 3(c). The detailed operational process is outlined in Algorithm 5 in Appendix (§A.1).

Algorithm 1 Object Detection and Box Interpolation-based Foreground Extraction

Require: $V = \{V_1, V_2, \ldots, V_N\}$ ▷ Set of input videos
Require: $\mathcal{M}_{\text{YOLO}}$, $\mathbf{w}$ ▷ YOLO model with weights
Require: $\mathcal{T}$ ▷ SORT tracker
Ensure: $V_{\text{fg}} = \{V_{1,\text{fg}}, V_{2,\text{fg}}, \ldots, V_{N,\text{fg}}\}$
Initialize $\mathcal{M}_{\text{YOLO}}$ with weights $\mathbf{w}$, load hyperparameters
for each $V_j \in V$ **do**
    Open V_j, retrieve FPS, (W, H)
    Initialize writer for Vj, fg
    Initialize mask $M_{j,i}$ as $\mathbf{0}^{H \times W}$
    for each frame $F_{j,i} \in V_j$ **do**
        1. Convert $F_{j,i}$ to RGB
        if $i \bmod 2 = 1$ **then**
            1. $\{O_k\}$, masks $\leftarrow \mathcal{M}_{\text{YOLO}}(F_{j,i})$
            2. $\{T_k\} \leftarrow \mathcal{T}(\{O_k\})$ ▷ Track detected objects
            3. Draw $\{T_k\}$ on $M_{j,i}$ and interpolate with $\{T_{k-1}\}$
        else
            Interpolate bounding boxes $\{T_{k-1}\}$ and draw on $M_{j,i}$
        end if
        2. $F_{j,i}[M_{j,i} = 0] \leftarrow (0, 255, 0)$ ▷ Set non-detected regions to green
        3. Write $F_{j,i}$ to $V_{j,\text{fg}}$
    end for
    Close writer for $V_{j,\text{fg}}$
end for

Segmentation and Trajectory Prediction To improve the speed of the instance segmentation model, the trajectories of each object are predicted using the SORT algorithm [11] including the Kalman filter. Due to the filter, the estimated velocities of objects are robust against noise. With the velocities, the next bounding boxes can be predicted as described in the following equation.

$$\mathbf{B}_t^i = \mathbf{B}_{t-1}^i + \mathbf{V}^i, \quad where\ \mathbf{V}^i = [v_x^i\ v_y^i\ v_w^i\ v_h^i]. \tag{2}$$

V^i is a variation of position and size of a bounding box. The segmentation model is applied to 3 frames and remembers the silhouette of the object. The silhouette is moved along the predicted trajectory for the next 2 consecutive frames. Figure 3(a) provides a visual explanation of this concept and the pseudo-code is detailed in Algorithm 5 in Appendix (§A.1). However, since the prediction is based on the assumption of constant velocity, trajectory prediction can be inaccurate when the object undergoes acceleration or rotational motion. As the video's frame rate decreases, this prediction error increases. In particular, as the objects are projected onto a 2D plane, their motion accelerates as they approach the camera. To compensate, the contour of the silhouette is thickened, shown in Figure 2(c) so that objects can still be captured on the scene even with prediction errors.

3.2 Background Extraction

To optimize storage efficiency in video surveillance, *CLOUD-CODEC* extracts a background image from a given video by estimating the static background across multiple frames. Since surveillance cameras primarily capture moving objects against a relatively stable background, storing entire video sequences results in redundant storage consumption. To mitigate this, our approach reconstructs a representative background image by processing a sequence of frames from the video.

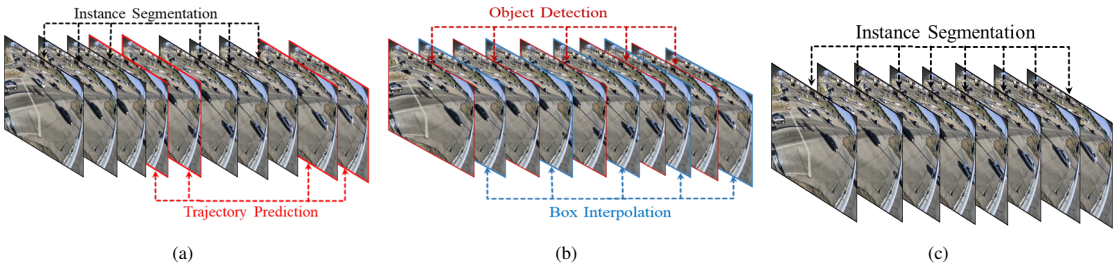


Fig. 3. Different Approaches to Reduce Computational Cost in *CLOUD-CODEC*. Model (a) reduces computational cost by skipping inference on two consecutive frames and applying trajectory prediction. Model (b) further optimizes by interpolating bounding boxes on every even frame. In contrast, model (c) involves heavy computation as it processes all frames without skipping.

The proposed background estimation technique relies on the pixel-wise median method, which determines the most representative background pixels over time. Specifically, for each pixel location, we collect pixel values from a randomly selected set of frames and compute their median value. This statistical approach ensures that transient foreground objects (e.g., vehicles, pedestrians) do not persist in the extracted background [48]. Algorithm 2 outlines the detailed background extraction concept.

3.3 Reconstruction

We restore the video by combining the foreground video and background image. The basic strategy is chroma-keying. When creating the foreground video, the RGB value of areas outside the object is set to (0, 255, 0), creating a green background. When combining with the background image, we remove the green background using chroma-keying and insert the image into the background.

Algorithm 2 Background Extraction

Require: $V = \{V_1, V_2, \ldots, V_N\}$ ▷ Input videos
Ensure: $B = \{B_1, B_2, \ldots, B_N\}$ ▷ Extracted backgrounds

for each video $V_j \in V$ **do**
  1.Open V_j, retrieve dimensions (W, H)
  2.Initialize empty list F_j for frames
  for each frame $F_{j,i}$ in V_j **do**
    1.Convert $F_{j,i}$ to grayscale
    2.Apply Gaussian blur: $F_{j,i,\text{gray}}$
    3.Add processed $F_{j,i}$ to F_j
  end for
  3.Apply Non-local Means Denoising on F_j
  4.Compute median of F_j to obtain background B_j
  5.Save B_j as the background image for video V_j
  6.Close video V_j
end for

Algorithm 3 Video Reconstruction

Require: $B = \{B_1, B_2, \ldots, B_N\}$ ▷ Backgrounds
Require: $V_{\text{fg}} = \{V_{1,\text{fg}}, \ldots, V_{N,\text{fg}}\}$ ▷ Foregrounds
Ensure: $V_{\text{reconstructed}} = V_{1,\text{re}}, V_{2,\text{re}}, \ldots, V_{N,\text{re}}$ ▷ Reconstructed

for each $V_{j,\text{fg}} \in V_{\text{fg}}$ **do**
  1.Load B_j for $V_{j,\text{fg}}$
  2.Open $V_{j,\text{fg}}$, retrieve (W, H) and FPS
  3.Initialize video writer for $V_{j,\text{re}}$
  for each frame $F_{j,i}$ in $V_{j,\text{fg}}$ **do**
    1.Define green thresholds $u_{\text{green}}, l_{\text{green}}$
    2.Create mask M_i for segmentation
    3.$F_{j,i,\text{re}} = B_j \odot (1 - M_i) + F_{j,i} \odot M_i$
    4.Write $F_{j,i,\text{re}}$ to $V_{j,\text{re}}$
  end for
  5.Close video writer and $V_{j,\text{fg}}$
end for

Algorithm 3 details this reconstruction approach. For each foreground video, a matching background image is loaded. Then, frame by frame, a binary mask isolates the object by filtering out non-object areas using green color thresholds. This mask helps segment the object from the green screen background, allowing the foreground object to be layered over the static background. Each frame is then reconstructed by merging the masked foreground with the background image, ensuring that only areas with detected objects are updated, while the background remains consistent. The result is a cohesive video that preserves the context of the original background and highlights the movements of relevant objects, optimizing storage by minimizing redundant background data.

4 IMPLEMENTATION AND DATASET

Backbone. We utilized *YOLOv7* with 36.9M parameters as the backbone model for the Deep Foreground Extractor. The pre-trained weights on the MS COCO dataset [38] were employed. We used only the object categories related to outdoor environments, including person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire hydrant, stop sign, and parking meter, while filtering out the remaining classes. The input frame size to the backbone was resized to 640 x 640 to obtain the positions of foreground objects. The confidence threshold was set to 0.2, and the IOU threshold was set to 0.45.

Dataset. We assess the performance of *CLOUD-CODEC* using a total of 145 surveillance videos obtained from various benchmarks, as summarized in Table 1. The detailed description of the datasets is as follows:

(1) **nVidia AI City Challenge** [44]: Published by NVIDIA, this dataset includes 100 videos featuring camera shivering, poor weather, and nighttime conditions. Video durations range from 19 seconds to 15 minutes, totaling 9 hours and 33 minutes. All videos are 1280×720, 10 FPS, H.264 encoded.
(2) **Bellevue** [46]: The Bellevue Traffic Video Dataset contains 101 hours of 1280×720, 30 FPS footage from five traffic intersections in Bellevue, Washington, recorded in September 2017. We randomly selected 10 videos for evaluation.
(3) **UA-DETRAC** [68]: We selected 20 video sequences from this benchmark dataset, which contains 100 traffic videos at 960×540, 30 FPS. Footage is captured at highways, intersections, and T-junctions in Beijing and Tianjin, China.
(4) **AAU RainSnow** [9]: This dataset focuses on traffic videos in rain and snow, collected from seven intersections in Aalborg and Viborg, Denmark. It includes RGB and thermal camera footage at 640×480, 20 FPS, captured from street lamp-mounted cameras. .

Encoding. In H.264 video encoding, we employ the default parameters of the FFmpeg libx264 library. It's true that the default parameter set of libx264 might not be the most optimal for all use cases. However, it provides a balance between video quality, file size, and encoding time. The default parameters of the H.264 encoding are outlined in Appendix (§A.3.)

Table 1. Summary of datasets used for evaluation

Dataset	# of Videos	Resolution	Duration	Type
AICC21	100	1280x720	19s-15 min	Benchmark
Bellevue	10	1280x720	1-25 min	Benchmark
DETRAC	20	960x540	1-2 min	Benchmark
AAU RainSnow	15	640x480	4-5 min	Benchmark

(a) High/low traffic volume (b) Pedestrian activity (c) Profound darkness (d) Early night/morning darkness

(e) Major intersections (f) Straight streets (g) Weather conditions (h) Severe conditions

Fig. 4. Examples of video captures over time, camera locations, traffic volume, pedestrian activity, and various weather conditions.

Dataset Diversity. The dataset varies across time periods, locations, traffic, weather, and lighting, as shown in Figure 4. Daytime videos have more objects and congestion, while nighttime videos show uniform object distribution. To ensure a diverse evaluation of *CLOUD-CODEC* and baselines, we selected 50 videos from 145, covering camera shaking, traffic variations, pedestrian activity, extreme darkness, intersections, straight roads, and severe weather conditions. Further details on these scenarios and video distribution. Further details on these scenarios and video distribution are provided in Appendix (§A.2).

Hardware. We have implemented and tested baseline deep foreground extraction (detection, segmentation, segmentation & prediction) and *CLOUD-CODEC* on Windows 11 machine and it has an Intel i9-10800 CPU with NVIDIA Titan RTX Graphics Card, and 128 GB memory. The videos are stored onto the local storage and streamed out to the baseline schemes and *CLOUD-CODEC*.

Baseline Schemes. As baselines, in addition to the detection, segmentation, segmentation & prediction-based foreground extraction modules, we have re-implemented `CloudSeg` [66] and an `AV1 video encoder`. `CloudSeg` is a model designed to perform real-time semantic segmentation by introducing the edge-to-cloud concept. The core idea is to reduce the resolution of streaming videos on the edge to increase inference speed and reduce network bandwidth. Then, apply super-resolution (SR) using the CARN model [1] to restore high-quality resolution from the low-resolution stream on the server side. To align with the requirements of `CloudSeg`, we apply a 2× bilinear down-sampling to all videos. Specifically, AICC21 and Bellevue videos are reduced from 1280×720 to 640×360, DETRAC videos from 960×540 to 480×270, and AAU RainSnow videos from 640×480 to 320×240. This technique—commonly used in VR streaming—reduces bandwidth usage by lowering quality during transmission, followed by enhancement on the user side [6, 7].

The AV1 video encoder performs compression on raw traffic surveillance footage, converting it to `AV1` format. The `AV1 video encoding` can achieve about 30% higher compression efficiency than VP9, and up to 50% higher efficiency than H.264 [14]. FFmpeg libaom-av1 library is used to implement the `AV1 video encoder` and its parameters are detailed in Appendix (§A.3.) After performing all required experiments in `CloudSeg` and `AV1 video encoder`, we were able to compare the performance of the baseline and *CLOUD-CODEC* based on below mentioned metrics.

Metrics. The performance evaluation considers the following five metrics:

(1) **Storage Size**. We report the comparative analysis of storage size based on baseline video compression and *CLOUD-CODEC* approaches to evaluate the encoding efficiency.
(2) **FPS**. FPS is used to measure the frame rate or how many frames of a video can be processed per second during encoding processes. FPS score serves as a key determinant of the system's real-time performance capabilities.
(3) **Video quality metrics.** We report three metrics—VMAF, SSIM, and PSNR—to assess how closely the decoded video matches the quality of the original source.
(4) **mAP Score**. We define the mAP score to evaluate the quality of decoded videos compared with original videos. The mAP (mean Average Precision) incorporates the ability of the detector to make correct classifications (precision) and the ability of the detector to find all relevant objects (recall) on the reconstructed videos with respect to ground truth (original videos).
(5) **Cost Analysis**. We demonstrate a cost analysis comparing the price of original video storing on bulk storage (HDD) versus encoding and storing videos with *CLOUD-CODEC* (GPU processing).

Evaluation Methodology. Our evaluation methodology encompasses three distinct strategies: first, we utilize four different benchmarks and evaluate them separately in our solution against the baselines. We asses storage saving (§5.1), encoding speed (§5.2), and video quality (§5.3) metrics, without considering the content of the videos and their variations across different conditions. Second, we use manually classified 50 videos to evaluate the impact of various conditions (camera shivering, traffic volume, weather, etc) on the performance of baselines and *CLOUD-CODEC* (§5.4). Lastly, we perform a cost analysis(§5.5) to determine which is cheaper: long-term bulk storage (HDD) or the GPU operational cost for *CLOUD-CODEC*-based video encoder.

5 EVALUATION RESULTS

5.1 Storage Saving

Figure 5 shows a comparison of *CLOUD-CODEC* and other baseline techniques across datasets in terms of encoded video size savings. The video size reduction results across the AICC21, Bellevue, DETRAC, and AAU datasets demonstrate varying levels of efficiency among the evaluated methods.

In the AICC21 dataset, the original video size is 4915 MB when encoded using the H.264 video format. On the other hand, the state-of-the-art AV1 video encoder demonstrated significantly better performance, compressing the same data to a size of 3092 MB which is 38% smaller than what was achieved using the H.264 encoder. After applying *CLOUD-CODEC* and other baseline approaches, the overall size of videos decreased remarkably. The detection, segmentation, and segmentation & prediction approaches could decrease video size up to 3.7×, 3.3×, and 2.8×, respectively. Similarly, in the Bellevue dataset, the original video size of 1140 MB is best encoded by *CLOUD-CODEC*. Other approaches like `AV1` and `CloudSeg` reduce the size up to 20% and 65%, respectively. In terms of DETRAC and AAU datasets, the original video sizes are reduced 4.5× and 6.2× using *CLOUD-CODEC*, respectively. Whereas, the `AV1` achieves 1.6× and 1.7× video size reduction rate, and `CloudSeg` reduces the original video sizes around 50% and 65% on these datasets.

Figure 6(a) shows that our proposed foreground-background extraction solutions achieve higher video size reduction rates compared to `AV1` and `CloudSeg` solutions. Specifically, Detection, Segmentation, Segmentation&Sort, and *CLOUD-CODEC* based video encoding solutions present moderate storage savings, exceeding 70% across the datasets. This significant storage saving is due to the smaller size occupied by foreground moving objects, whereas, the background static objects significantly influence the video's overall size. Among the foreground-background extraction approaches

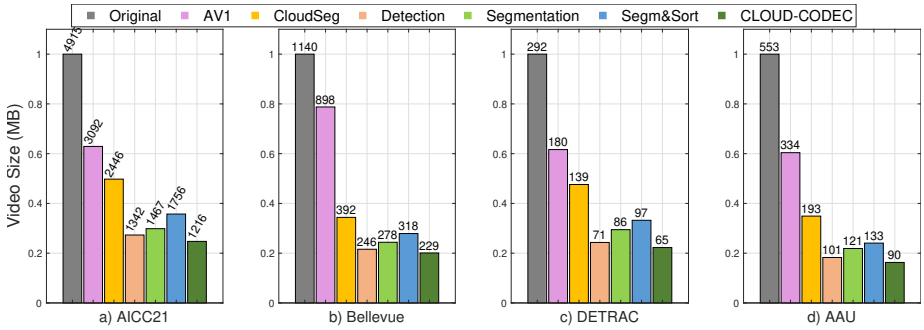


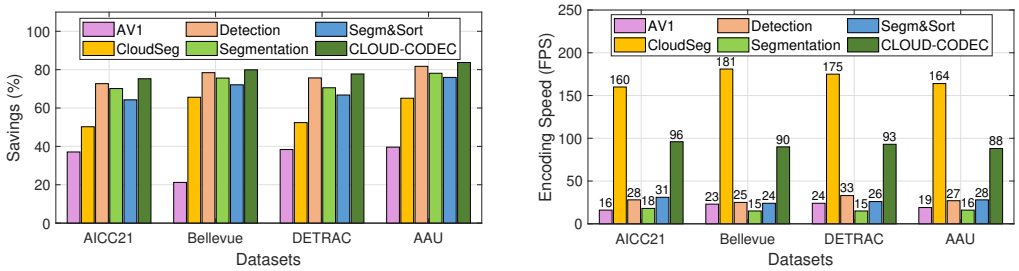
Fig. 5. Encoded Video Size Comparison of *CLOUD-CODEC* with Alternative Methods across datasets.

CLOUD-CODEC dominates in achieving the highest video size reduction rates across all datasets. The reason why a foreground video in the shape of a bounding box is estimated to achieve the best size reduction may be due to the block-based motion compensation used in H.264 [69]. *CLOUD-CODEC* encodes the video in the H.264 format, just like the original video, right before storage. During this encoding process, the block size and number are determined while performing motion compensation. Because the shape of the bounding box is simpler than the exact contour of the object, fewer blocks may be required for the calculation, leading to greater storage savings.

5.2 Encoding Speed

We now measure the encoding speed of *CLOUD-CODEC* and competing approaches on the datasets. The FPS metric is employed to evaluate the encoding speed. Figure 6(b) represents the comparative analysis of the average FPS rate of the mentioned techniques over the four datasets. As can be seen, the `AV1` video encoder achieves one of the lowest FPS rates and it can encode 16, 23, 24, and 19 frames per second on AICC21, Bellevue, DETRAC, AAU RainSnow datasets, respectively. While the segmentation-assisted video encoder exhibits consistently lower FPS (between 15 and 18) due to higher computational demand in the foreground extraction processes. Wheres, the detection and segmentation&sort-based foreground extraction approaches encode on average 28 fps over the datasets, respectively. The proposed *CLOUD-CODEC* based encoding approach has outperformed with 96, 90, 93, and 88 fps throughout the datasets among the other foreground extraction approaches. According to the FPS score, we can claim that the proposed *CLOUD-CODEC* is highly acceptable for real-time video encoding. It is true that, `CloudSeg` achieves the better FPS rate compared to the *CLOUD-CODEC* across all datasets, this is because it only down-samples the video quality to the resolution of 0.5× the original.

Table 3 in Appendix (§A.4) presents the statistical summary of the video size reduction rates (%) across different datasets for *CLOUD-CODEC* and competing methods. The results highlight that *CLOUD-CODEC* consistently outperforms all baseline approaches across the four datasets. Specifically, for the Bellevue and AAU datasets, *CLOUD-CODEC* achieves the highest compression rates of 80.3% and 82.4%, respectively, surpassing detection-based methods, which were the second-best performers. Similarly, for the AICC21 and DETRAC datasets, *CLOUD-CODEC* reduces video sizes by 74.9% and 78.8%, respectively, demonstrating its effectiveness across diverse datasets. The 95 % confidence intervals, Cl Upper and Cl Lower, demonstrate the statistical stability of the method, further validating its reliability in practical scenarios.



(a) Storage savings (b) Encoding speed

Fig. 6. Comparative analysis of storage savings and encoding speed.

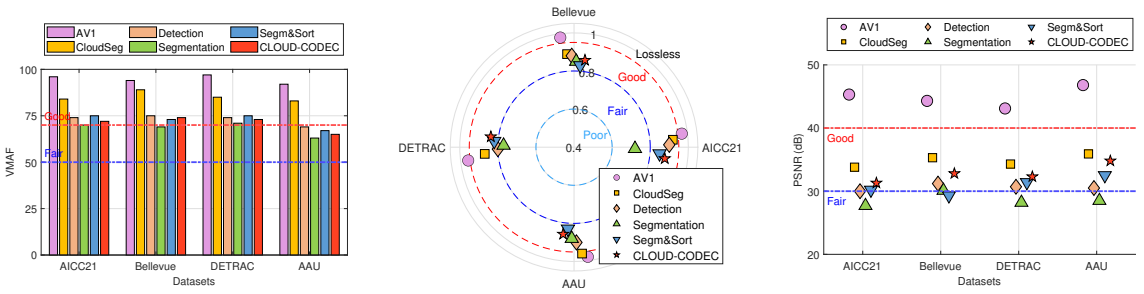
5.3 Quality of Reconstructed Videos

5.3.1 Analysis of video quality. As mentioned above, the foreground moving objects and background scenes are combined in the reconstruction phase to decode the full scene. We evaluated the reconstructed video quality using VMAF (v0.6.1), SSIM, and PSNR, leveraging FFmpeg for automated computation. VMAF, SSIM, and PSNR were computed using FFmpeg to evaluate reconstructed video quality. VMAF was obtained via the *libvmaf* filter [45], assessing perceptual quality by comparing reconstructed videos to their originals. SSIM and PSNR were calculated using the *ssim* and *psnr* filters [25], measuring structural similarity and pixel-level distortion in decibels (dB), respectively. The descriptions of these metrics are as follows:

(1) **VMAF:** The VMAF score ranges from 0 to 100, with higher scores indicating better video quality based on human subjective ratings [37]. A score below 50 is poor, 50–70 is fair, 70–90 is good, and 90+ is excellent. A VMAF score of 70 indicates a transition between "fair" and "good" quality.
(2) **SSIM**: Structural Similarity Index evaluates structural details, luminance, and contrast, ranging from 0 to 1, where higher values indicate greater similarity. Scores below 0.6 indicate poor quality, 0.6–0.8 suggests noticeable degradation, 0.8–0.95 is good quality, and above 0.95 is nearly identical to the original.
(3) **PSNR:** Peak Signal-to-Noise Ratio (PSNR), measured in decibels (dB), quantifies pixel-level differences. A PSNR below 20 dB suggests poor quality, 20–30 dB indicates moderate quality with visible artifacts, 30–40 dB represents good quality, and above 40 dB is nearly lossless.

Figure 7 presents a comparative analysis of video quality metrics for *CLOUD-CODEC* and baseline techniques across all datasets. In Figure 7(a), detection, segmentation, Segm&Sort prediction, and *CLOUD-CODEC* achieved VMAF scores of 74, 70, 76, and 73 on AICC21, all classified as “good” (VMAF > 70). `AV1` (96) and `CloudSeg` (83.5) outperformed all methods. As validated by previous studies [78], `AV1` is the top codec, offering the same perceived quality as H.265 while using 12% less data. `CloudSeg` retains all foreground and background scenes, resulting in high VMAF. Foreground extraction-based methods generally maintain "good" quality, though segmentation slightly underperforms on Bellevue and DETRAC datasets. The AAU RainSnow dataset, with rain, snow, and night conditions, presents greater challenges, where only `AV1` and `CloudSeg` maintain high quality, while foreground extraction methods are rated as "fair."

Figure 7(b) assesses structural similarity (SSIM) between reconstructed and original videos, emphasizing luminance, contrast, and structural integrity. `AV1` achieves SSIM values of 0.96–0.98, preserving near-perfect structure, while CloudSeg, Detection, Segmentation, and Segm&Sort range from 0.72 to 0.96. Some methods exhibit moderate quality on



(a) VMAF (b) SSIM (c) PSNR

Fig. 7. Reconstructed video quality comparison of *CLOUD-CODEC* and baseline methods using (a) VMAF, (b) SSIM, and (c) PSNR metrics.

difficult datasets. *CLOUD-CODEC* consistently falls in the "good" to "lossless" range, balancing quality and storage efficiency.

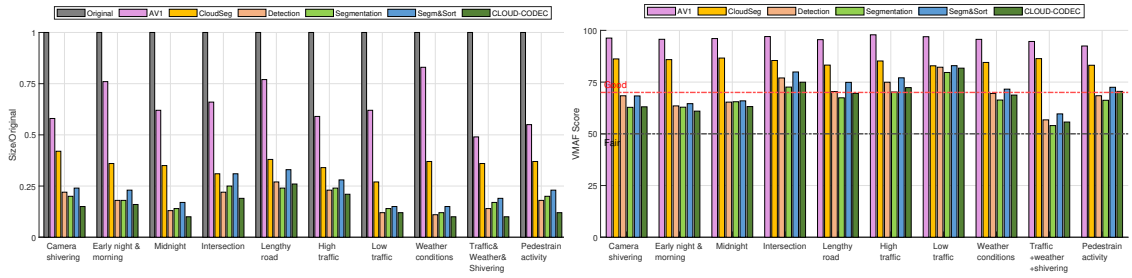
Figure 7(c) analyzes pixel-level differences using Peak Signal-to-Noise Ratio (PSNR). Our approach enhances storage efficiency by separating foreground from background, storing only dynamic objects while keeping the static background as an image. During reconstruction, the video is regenerated by overlaying the foreground onto the stored background. While effective for compression, this method may reduce PSNR since the reconstructed video does not perfectly match original frames. `AV1` consistently achieves the highest PSNR, retaining full pixel data, while `CloudSeg` also performs well by preserving both foreground and background. Foreground extraction methods, including *CLOUD-CODEC*, yield moderate PSNR values (Fair: 20–30 dB, Good: 30–40 dB), reflecting minor pixel inconsistencies due to background substitution.

5.3.2 Analysis of mAP score. To address how well the reconstructed videos preserve foreground information, we adopt the mAP score in deep foreground extraction. This involves YOLOv8m-based object detection [49] on both the original and the reconstructed videos. We consider the object detection results from original videos as the *ground truth*-standard for comparison. Then, we calculate the mAP scores for the reconstructed videos, focusing on how closely object detection results match the ground truth from the original videos.

Table 2 summarizes the overall performance of the detection similarity between the original and reconstructed videos across different datasets and encoding solutions. As can be seen, the mAP score of the object detection on both original and reconstructed videos is remarkably high in `AV1` throughout datasets, and it consistently outperforms other encoding solutions. This strong performance is attributed to `AV1`'s advanced encoding techniques, which ensure that the encoded

Table 2. Average Precision Scores across Different Datasets and Solutions

Datasets	**AICC21**			**Bellevue**			**Detrac**			**AAU**		
Solutions	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
`AV1`	0.89	0.94	0.93	0.90	0.93	0.92	0.90	0.94	0.93	0.89	0.91	0.92
`CloudSeg`	0.84	0.90	0.89	0.82	0.89	0. 85	0.87	0.92	0.91	0.83	0.91	0.90
Detection	0.87	0.93	0.93	0.86	0.91	0.90	0.89	0.92	0.91	0.79	0.82	0.80
Segmentation	0.84	0.93	0.92	0.82	0.92	0.91	0.80	0.93	0.92	0.75	0.81	0.79
Segmentation & Sort	0.87	0.93	0.93	0.84	0.90	0.89	0.87	0.91	0.90	0.75	0.84	0.81
CLOUD-CODEC	0.85	0.93	0.92	0.84	0.92	0.91	0.85	0.93	0.92	0.76	0.82	0.80



(a) Storage Saving (b) Reconstructed Video Quality (VMAF)

Fig. 8. Performance analysis under various conditions.

videos are nearly identical to the original ones in terms of visual quality and detail. Whereas, `CloudSeg` demonstrates lower mAP scores over the datasets compared to the other approaches. The resolution and detail of the videos are significantly reduced due to `CloudSeg`'s downsampling method, impacting the object detection model's accuracy in detecting and localizing objects.

In deep foreground extraction-based encoding solutions, we observe nearly consistent mAP scores across the AICC21, Bellevue, and Detrac datasets. This consistency indicates a high degree of similarity between the original and reconstructed videos, enabling the object detection algorithm to perform effectively. Note that the algorithm struggles to distinguish between the original and reconstructed videos, highlighting the robustness of the encoding process in preserving important foreground information. However, the AAU Rain-Snow dataset comprises challenging environmental conditions that significantly affect foreground extraction and the quality of the reconstructed videos. Factors such as rain, snow, and nighttime scenes reduce the clarity and visibility of foreground objects, leading to decreased mAP scores on this dataset.

5.4 Impact of Various Conditions

To evaluate *CLOUD-CODEC* robustness, we consider various conditions over time, camera locations, traffic volume, weather conditions, and pedestrian activity as described in §4. Figure 8 summarizes the overall size reduction and VMAF score analysis under diverse scenarios. Below, we discuss our findings in detail according to each condition.

Camera shivering. Camera shivering indirectly affects video size reduction as compression algorithms (e.g., H.264, H.265) rely on frame-to-frame redundancies. Increased frame variations due to shaking lead to larger file sizes. *CLOUD-CODEC* achieves an 85% size reduction, outperforming `CloudSeg` (58%) and `AV1` (42%). Other foreground extraction methods—detection (76%), segmentation (81%), and segmentation & sort (70%)—offer varying levels of efficiency. In terms of quality, DFE-based decoded videos range from "fair" to "good" (VMAF: 62–70) under camera-shivering conditions, whereas `CloudSeg` and `AV1` achieve significantly higher scores of 85 and 95, respectively.

Midnight vs early night/morning. The time of the day and lighting conditions significantly affect CCTV video sizes. The `AV1` achieves 24% and 38% compression for early night/morning and midnight videos, respectively. `CloudSeg` maintains a 65% encoding rate under both conditions, while *CLOUD-CODEC* outperforms both, achieving 84% for early night/morning and 91% for midnight videos. Although original video sizes remain similar across both conditions, foreground extraction methods (including *CLOUD-CODEC*) demonstrate significantly better compression than `CloudSeg` and `AV1`. Since early night/morning has more moving objects, foreground extraction achieves an even higher encoding

rate for midnight videos, where static backgrounds dominate. Regarding decoded video quality, both conditions yield similar ratings (VMAF: 62–68). Overall, *CLOUD-CODEC* proves highly effective for nighttime surveillance, offering superior encoding efficiency and quality retention.

Intersection vs Lengthy road. We analyzed encoding rates and decoded video quality based on camera placement. For major intersections, `AV1` and `CloudSeg` had the lowest encoding rates at 35% and 68.7%, while detection, segmentation, and segmentation&sort achieved 77%, 75%, and 69%, respectively. *CLOUD-CODEC* led with 81%, demonstrating superior compression efficiency. For lengthy roads, `CloudSeg` encoded at 62%, other baselines averaged 72%, and *CLOUD-CODEC* reached 75%. Decoded video VMAF scores ranged from 72 to 80 for intersection videos and 70 for lengthy roads. `AV1` and `CloudSeg` provided the highest quality, while *CLOUD-CODEC* balanced strong encoding efficiency with competitive quality across camera placements.

High vs Low traffic. We compared encoding efficiency and decoded video quality under high and low traffic conditions, where a video is classified as high traffic if a single frame contains more than 15 moving objects. `CloudSeg` achieved compression rates of 66.5% and 71% for high and low-traffic videos, respectively, while *CLOUD-CODEC* outperformed with 81% and 89% file size reduction. Among other foreground extraction methods, only the detection-based approach achieved a comparable encoding rate across both conditions. For video quality, low-traffic videos yielded higher VMAF scores, as fewer moving objects resulted in less compression loss. However, both high and low-traffic videos maintained "good" viewer ratings, demonstrating *CLOUD-CODEC*'s ability to balance efficient compression with quality retention.

Weather and Severe conditions. We assessed *CLOUD-CODEC*'s performance across various weather conditions (rain, snow, fog) and mixed severe conditions (high traffic, camera shivering, rain or snow). Detection, segmentation, and segmentation & sort methods achieved 85% size reduction, while `CloudSeg` and *CLOUD-CODEC* reached 63% and 90%, respectively. Decoded videos maintained a VMAF score of 70, indicating good quality across different conditions. `AV1` struggled in weather-affected scenarios, achieving only 18% compression due to frame variations. While *CLOUD-CODEC* and other baselines achieved similar size reductions, VMAF scores were notably lower under severe conditions due to increased motion, camera instability, and extreme weather effects.

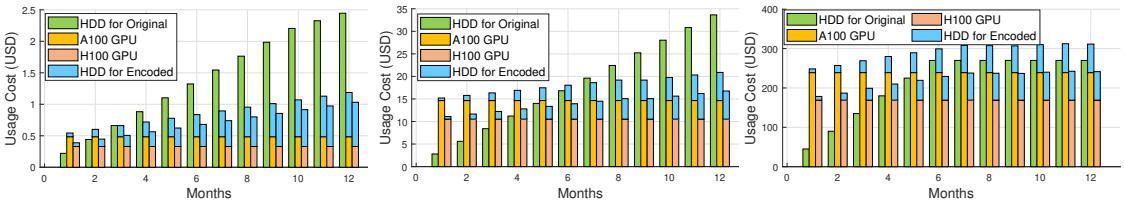
Pedestrian activity. We analyzed five recordings where pedestrian density exceeds vehicle density. `AV1` reduced video size to 45%, while `CloudSeg` and detection achieved 67% and 82%, respectively. Segmentation-based methods showed similar efficiency, and *CLOUD-CODEC* achieved the highest compression at 90%. For video quality, `AV1` scored highest with a VMAF of 92.45, followed by `CloudSeg` at 83.14. Detection and segmentation methods yielded moderate VMAF scores, slightly below the "good" threshold. Segmentation & Sort (72.5) and *CLOUD-CODEC* (70.5) maintained "better" quality ratings, balancing compression and visual fidelity.

5.5 Cost Analysis

We now discuss a cost analysis by comparing two approaches for storing traffic surveillance videos. The first approach involves directly storing the original videos on bulk storage (HDD), while the other uses GPU-based processing with the *CLOUD-CODEC* method to encode the videos before storage. Appendix (§A.5) details the HDD and GPU service providers and their prices.

We evaluate two scenarios:

(1) *Static Storage*: The dataset size remains constant over 12 months. The AICC21 and Bellevue datasets are selected due to their large-scale video data. AICC21 (4.9 GB original, 1.3 GB encoded) requires 0.47 hours of A100 GPU processing and 0.17 hours on an H100. Bellevue (62.3 GB original, 12.6 GB encoded) requires 14.03 hours on an



(a) AICC21 (b) Bellevue (c) Incremental

Fig. 9. HDD and GPU usage cost analysis. The datasets AICC21 9(a) and Bellevue 9(b) are used for static data storing scenario, while figure 9(c) shows the incremental storing scenario.

A100 and 5.27 hours on an H100. Smaller datasets (UA-DETRAC and AAU RainSnow) are excluded due to their lower resolution and size.

(2) *Incremental Storage*: Data grows by 1,000 GB each month, starting from 1,000 GB, with an average encoded size of 210 GB. GPU processing times increase with data size, averaging 230 hours on an A100 and 84.6 hours on an H100. To manage storage, videos older than six months are deleted.

Table 5 in Appendix (§A.5) summarizes the processing time and video size results before and after encoding. Figure 9 presents a cost analysis of HDD and GPU usage across three datasets over 12 months. Across all cases, H100 usage costs are about 30% lower than A100 due to reduced processing time. Figures 9(a) and 9(b) illustrate static storage scenarios for AICC21 and Bellevue datasets. Storing the 4.9 GB AICC21 dataset on Amazon Optimized HDD costs 0.22, 0.44, 0.66, and 0.88 USD for 1 to 4 months. Encoding it to 1.3 GB costs 0.48 (A100) and 0.33 (H100), with the encoded dataset incurring storage costs of 0.05, 0.11, 0.17, and 0.234 USD over the same period. If stored for more than three months, *CLOUD-CODEC* achieves higher cost efficiency, and at six months, H100-assisted encoding is 2× cheaper than storing original videos on HDD. For the 62.3 GB Bellevue dataset, encoding costs 14.6 (A100) and 10.5 (H100) per month, while storing the unencoded dataset ranges from $3 to 14$ in the first five months. Beyond five months, encoding & storing with *CLOUD-CODEC* consistently reduces costs compared to raw HDD storage.

In the incremental storage scenario, where data grows by 1,000 GB per month, initial H100 GPU processing costs are 3.5× higher than HDD storage. At 2 and 3 months, GPU costs drop to 2× and 1.4× higher than HDD, respectively. However, for longer retention (4+ months), *CLOUD-CODEC* becomes more cost-effective. Although GPU costs are higher in shorter retention periods (0–4 months), the long-term savings justify encoding overhead. Appendix (§A.5) highlights additional insights supporting our claim.

6 DISCUSSIONS

6.1 Challenging Scenes

The system introduced in this paper is improved and more effective as compared to the pre-existing approaches. Our proposed approach ensures real-time performance and efficient storage space. However, due to certain limitations, it is not applicable in challenging conditions where details are crucial. In this section, we describe these limitations and discuss possible solutions to address them.

Residual Vehicle Lights in Night Scene Although *CLOUD-CODEC* performs well with a moderately good VMAF score for dark conditions as mentioned in Section 5.4, some results have certain limitations as depicted in Figure 10. Here the

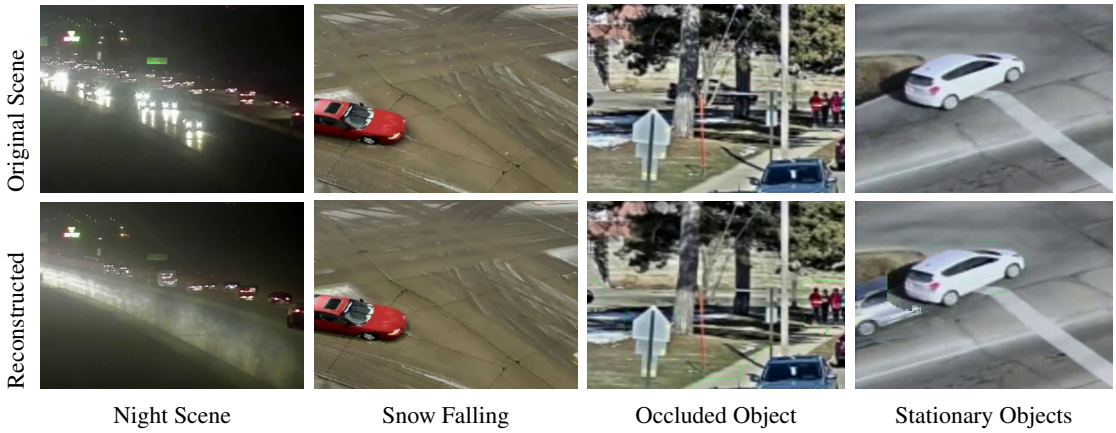


Fig. 10. Challenging Conditions.

vehicle's lights seem to leave light traces in the background image. This can be reduced by referring to the research of flare removal [40, 72]. Additionally, a lesser number of vehicles were detected compared to the count in daytime videos. This is because the model has not been trained for nighttime videos. If the model is re-trained with the night scenes, it is expected to improve the current performance.

Outliers in Weather Conditions. After decoding all video files, we conducted a manual inspection to assess the quality of the reconstructed videos. This review identified a few outliers in severe weather conditions. For example, when the original footage depicted snowfall, the reconstructed video often failed to capture falling snowflakes accurately, as shown in Figure 10. While retraining the model to detect snowfall, rain, and other weather elements could enhance video quality, it would also increase file size. To balance quality and storage efficiency, we classified these weather-related discrepancies as outliers and accepted the current limitations to maintain optimal compression.

Occluded Objects Most videos in our dataset contain identifiable objects, but some include occluded objects, which may be critical or insignificant depending on the context. Our system relies on object detection algorithms, which can detect most occlusions. However, when occlusion exceeds a certain threshold, detection may fail, as these algorithms focus on visible objects per frame and may overlook occluded ones. To address this, integrating tracking algorithms [70, 77] with *YOLOv7* can enhance occlusion handling. Unlike standalone *YOLOv7*, tracking methods maintain object consistency across frames, improving detection and reducing missed occlusions.

Objects in Long Distance The proposed approach might achieve low detection performance for distant objects. While using a larger model that can handle high-resolution inputs is possible, it may eliminate the advantage of real-time processing due to increased computational cost. There are studies that have addressed this issue using modified *YOLOv3* [41] or Mixture of Gaussian (MoG) model and Variable Weighted Pipeline Filter [73]. Through these studies, we can improve the performance of this aspect in the future.

Untrained Objects The biggest disadvantage of object detection is that it cannot detect objects that have not been trained. Moreover, even with trained objects, it may fail to detect them depending on their shape. As a solution, there are methods such as applying unsupervised learning-based models or using ensemble with the results of other models. Another way is to utilize zero-shot learning-based models. More research is needed to apply these methods to our system while ensuring real-time performance.

Ambiguous Background Criteria Distinguishing background elements can be challenging, especially when stationary objects are later in motion, potentially leaving afterimages in reconstructed videos— a common limitation of dynamic background extraction. To address this, leveraging weeks of contextual data helps replenish excluded pixels once objects move, while prior background frames can replace pixels for long-term stationary objects. Additionally, shadows, glare, and stationary objects may obscure road markings. HomoFusion [65] can extract complementary cues from adjacent frames, improving background modeling and preserving occluded road features.

Challenges in Dynamic Camera Environments Our study focuses on static cameras, which maintain a fixed position and are widely used in traffic surveillance for their reliability. However, dynamic cameras (e.g., PTZ models or mobile setups) pose challenges for foreground and background extraction due to shifting scenes. Addressing these conditions requires adaptive background models that adjust to changing viewpoints, potentially using multiple models or adaptive learning techniques. Future work will explore these methods to improve adaptability in motion-capable camera environments.

6.2 Mitigating Information Loss Risks

Due to the limitations mentioned above, it is impractical to apply the proposed system even though the compression rate is high. This is because the information judged as the background and removed by the system can be important. In particular, background information plays a crucial role in analyzing video footage of traffic accidents. For this reason, instead of directly applying this system to video data, we suggest applying it only to occasional data after going through abnormal traffic analysis [53, 63]. Additionally, we recommend storing the data as is for a certain period, and then applying this system only for long-term storage to mitigate any potential risks occurring by information loss.

6.3 Trade-off between Quality and Size

As shown in Figure 6, `AV1` and `CloudSeg` have higher VMAF scores but less size reduction rate compared to the foreground videos. The trade-off between quality and file size is an inevitable aspect when discussing video compression. AV1 and `CloudSeg` maintain high quality by preserving background information almost identically to the original, resulting in lower compression rates. In contrast, foreground videos dramatically reduce in size by eliminating static background information, but this leads to a relatively lower quality due to the loss of background information. *CLOUD-CODEC* prioritizes the importance of foreground, maintaining quality in this aspect while accepting loss in the background to achieve higher compression rates.

6.4 Broader Applicability of *CLOUD-CODEC*

In addition to optimizing surveillance storage, CLOUD-CODEC has the potential to significantly impact various urban computing applications. For instance, in infrastructure maintenance, it can process crowdsourced dashcam footage to detect pavement distresses, enhancing road safety and maintenance efficiency [17]. In intelligent transportation systems, CLOUD-CODEC enables real-time traffic analysis, supporting improved traffic management and congestion reduction [26].

Furthermore, CLOUD-CODEC preserves essential visual characteristics while optimizing storage efficiency. We evaluated its visual quality using VMAF, PSNR, and SSIM, all of which indicate high-fidelity visual representation. This ensures that compressed videos retain key spatial, temporal, and object-level details, making them valuable for transportation analytics, infrastructure assessment, and environmental monitoring. These capabilities reinforce CLOUD-CODEC's role beyond surveillance, positioning it as a versatile tool for broader urban computing research.

7 FUTURE WORKS

CLOUD-CODEC has potential applications beyond traffic surveillance, such as general video storage scenarios. Implementing *CLOUD-CODEC* can lead to significant storage size reductions (Figure 11), up to 74.92%. The system can be adapted to process concurrent surveillance videos from multiple cameras, further improving storage efficiency. With the integration of server-level GPUs, the performance of CLOUD-CODEC can be significantly enhanced, as object detection algorithms will benefit from faster processing. As better GPUs become more affordable, this will become increasingly feasible. Combining *CLOUD-CODEC* with motion-sensor cameras can optimize storage use by only capturing frames containing moving objects, instead of recording continuously. This approach can save considerable storage space. By incorporating *CLOUD-CODEC* directly into cameras' firmware could generate a single background image and a foreground video with a much smaller file size. This would reduce the bandwidth needed to send data to the server, improving overall efficiency.

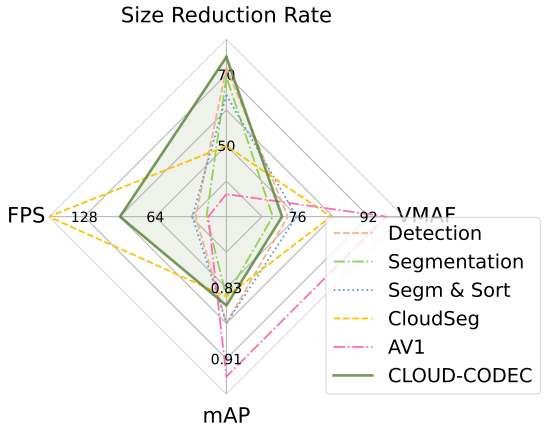


Fig. 11. Comparison of Overall Performance

8 CONCLUSION

CLOUD-CODEC offers a promising solution for reducing storage space and management costs in traffic surveillance systems by effectively separating video data into foreground and background. While the current approach may not capture extremely abnormal events in detail, the adoption of the bounding box with the interpolation method allows for real-time performance and efficient processing of numerous video streams. Future advancements in foreground extraction algorithms may further enhance the system's robustness and processing speed, making *CLOUD-CODEC* even more effective.

ACKNOWLEDGMENTS

This work was supported by the Ministry of Science and ICT (MSIT) through the ICT Creative Consilience program (IITP-2019-2011-1-00783), and by the Ministry of Land, Infrastructure and Transport (MOLIT) through the K-Tunnel project titled Development of Technology to Enhance Safety and Efficiency of Ultra-Long K-Underground Expressway Infrastructure (RS-2024-00416524). Arani Bhattacharya's portion of this work was additionally supported by the Cisco

University Research Fund, Cisco Grant number 76417363 (SVCF Grant 2022-318921). The authors sincerely thank the anonymous reviewers and editors for their insightful comments and constructive suggestions.

REFERENCES

[1] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. 2018. Fast, accurate, and lightweight super-resolution with cascading residual network. In *Proceedings of the European conference on computer vision (ECCV)*. 252–268.

[2] Aftab Alam, Irfan Ullah, and Young-Koo Lee. 2020. Video big data analytics in the cloud: A reference architecture, survey, opportunities, and open research issues. *IEEE Access* 8 (2020), 152377–152422.

[3] Panteha Alipour and Asadollah Shahbahrami. 2022. An adaptive background subtraction approach based on frame differences in video surveillance. In *2022 International Conference on Machine Vision and Image Processing (MVIP)*. 1–5. https://doi.org/10.1109/MVIP53647.2022.9738762

[4] AWS. 2024. *Amazon EBS pricing*. https://aws.amazon.com/ebs/pricing/?did=ap_card&trk=ap_card 2024-09-10.

[5] AWS. 2024. *Amazon EC2 Capacity Blocks for ML Pricing*. https://aws.amazon.com/ec2/capacityblocks/pricing/ 2024-09-10.

[6] Duin Baek, Mallesham Dasari, Samir R. Das, and Jihoon Ryoo. 2021. dcSR: practical video quality enhancement using data-centric super resolution. In *2021 CoNEXT*. ACM, 336–343.

[7] Duin Baek, Hangil Kang, and Jihoon Ryoo. 2020. SALI360: design and implementation of saliency based video compression for 360° video streaming. In *2020 MMSys*. ACM, 141–152.

[8] Duin Baek, Youngchan Lim, and Jihoon Ryoo. 2024. SenseQ: Context-Aware Video Quality Adaptation for Optimal Mobile Video Streaming in Dynamic Environments. *IEEE Access* 12 (2024), 20209–20220.

[9] Chris H. Bahnsen and Thomas B. Moeslund. 2018. Rain Removal in Traffic Surveillance: Does it Matter? *IEEE Transactions on Intelligent Transportation Systems* (2018), 1–18. https://doi.org/10.1109/TITS.2018.2872502

[10] Kamel Belloulata, Amina Belalia, and Shiping Zhu. 2014. Object-based stereo video compression using fractals and shape-adaptive DCT. *AEU-International Journal of Electronics and Communications* 68, 7 (2014), 687–697.

[11] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. 2016. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*. 3464–3468. https://doi.org/10.1109/ICIP.2016.7533003

[12] Shubham Chaudhary, Aryan Taneja, Anjali Singh, Purbasha Roy, Sohum Sikdar, Mukulika Maity, and Arani Bhattacharya. 2024. {TileClipper}: Lightweight Selection of Regions of Interest from Videos for Traffic Surveillance. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. 967–984.

[13] Fangdong Chen, Houqiang Li, Li Li, Dong Liu, and Feng Wu. 2016. Block-composed background reference for high efficiency video coding. *IEEE Transactions on Circuits and Systems for Video Technology* 27, 12 (2016), 2639–2651.

[14] Yue Chen, Debargha Mukherjee, Jingning Han, Adrian Grange, Yaowu Xu, Sarah Parker, Cheng Chen, Hui Su, Urvang Joshi, Ching-Han Chiang, et al. 2020. An overview of coding tools in AV1: The first video codec from the alliance for open media. *APSIPA Transactions on Signal and Information Processing* 9 (2020), e6.

[15] Zhenghao Chen, Guo Lu, Zhihao Hu, Shan Liu, Wei Jiang, and Dong Xu. 2022. LSVC: a learning-based stereo video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6073–6082.

[16] Der-Chun Cherng and Shao-Yi Chien. 2007. Video segmentation with model-based sprite generation for panning surveillance cameras. In *2007 IEEE International Symposium on Circuits and Systems*. IEEE, 2902–2905.

[17] Bahar Dadashova, Chiara Silvestri Dobrovolny, Mahmood Tabesh, Safety through Disruption, et al. 2021. *Detecting pavement distresses using crowdsourced dashcam camera images*. Technical Report. Safety through Disruption (Safe-D) University Transportation Center (UTC).

[18] Prasanga Dhungel, Prashant Tandan, Sandesh Bhusal, Sobit Neupane, and Subarna Shakya. 2020. Video Compression for Surveillance Application using Deep Neural Network. *Journal of Artificial Intelligence and Capsule Networks* 2, 2 (2020), 131–145.

[19] Dandan Ding, Junchao Tong, and Lingyi Kong. 2020. A deep learning approach for quality enhancement of surveillance video. *Journal of Intelligent Transportation Systems* 24, 3 (2020), 304–314.

[20] Dongna Du, Chenghao Zhang, Yanbo Wang, Xiaoyun Kuang, Yiwei Yang, Kaitian Huang, and Kejie Huang. 2021. A Compression and Recognition Joint Model for Structured Video Surveillance Storage. In *International Conference on Frontiers of Electronics, Information and Computation Technologies*. 1–8.

[21] Kuntai Du, Ahsan Pervaiz, Xin Yuan, Aakanksha Chowdhery, Qizheng Zhang, Henry Hoffmann, and Junchen Jiang. 2020. Server-driven video streaming for deep learning inference. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 557–570.

[22] Kuntai Du, Qizheng Zhang, Anton Arapin, Haodong Wang, Zhengxu Xia, and Junchen Jiang. 2022. Accmpeg: Optimizing video encoding for video analytics. *arXiv preprint arXiv:2204.12534* (2022).

[23] Omar Elharrouss, Noor Almaadeed, and Somaya Al-Maadeed. 2021. A review of video surveillance systems. *Journal of Visual Communication and Image Representation* 77 (2021), 103116.

[24] Aleksandr Fedorov, Kseniia Nikolskaia, Sergey Ivanov, Vladimir Shepelev, and Alexey Minbaleev. 2019. Traffic flow estimation with data from a video surveillance camera. *Journal of Big Data* 6 (2019), 1–15.

[25] FFmpeg Developers. 2025. *FFmpeg Filters Documentation*. https://ffmpeg.org/ffmpeg-filters.html Accessed: 2025-02-19.

[26] Matt Franchi, Debargha Dey, and Wendy Ju. 2024. Towards Instrumented Fingerprinting of Urban Traffic: A Novel Methodology using Distributed Mobile Point-of-View Cameras. In *Proceedings of the 16th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. 53–62.

[27] Hongpeng Guo, Shuochao Yao, Zhe Yang, Qian Zhou, and Klara Nahrstedt. 2021. CrossRoI: cross-camera region of interest optimization for efficient real time video analytics at scale. In *Proceedings of the 12th ACM Multimedia Systems Conference*. 186–199.

[28] Xiaoming Guo, Guang Jiang, Zhaopeng Cui, and Pei Tao. 2016. Homography-based block motion estimation for video coding of PTZ cameras. *Journal of Visual Communication and Image Representation* 39 (2016), 164–171.

[29] Marius Hobbhahn and Tamay Besiroglu. 2022. Trends in GPU Price-Performance. https://epochai.org/blog/trends-in-gpu-price-performance Accessed: 2024-10-28.

[30] HPC-AI.COM. 2024. *Accelerating Al to power the future of Intelligence*. https://hpc-ai.com/ 2024-10-27.

[31] Zinah N Idan, Sadiq H Abdulhussain, Basheera M Mahmmod, Khaled A Al-Utaibi, Syed Abdul Rahman Al-Hadad, and Sadiq M Sait. 2021. Fast shot boundary detection based on separable moments and support vector machine. *IEEE Access* 9 (2021), 106412–106427.

[32] Yeondae Jung and Andrew P Wheeler. 2023. The effect of public surveillance cameras on crime clearance rates. *Journal of Experimental Criminology* 19, 1 (2023), 143–164.

[33] Shailesh Kamble, Nileshsingh Thakur, and Preeti Bajaj. 2017. Modified three-step search block matching motion estimation and weighted finite automata based fractal video compression. (2017).

[34] Ashutosh Kumar, Takehiro Kashiyama, Hiroya Maeda, Hiroshi Omata, and Yoshihide Sekimoto. 2022. Real-time citywide reconstruction of traffic flow from moving cameras on lightweight edge devices. *ISPRS Journal of Photogrammetry and Remote Sensing* 192 (2022), 115–129.

[35] Julian Laufs, Hervé Borrion, and Ben Bradford. 2020. Security and the smart city: A systematic review. *Sustainable cities and society* 55 (2020), 102023.

[36] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. 2020. Reducto: On-camera filtering for resource-efficient real-time video analytics. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 359–376.

[37] Zhi Li, Christos Bampis, Julie Novak, Anne Aaron, Kyle Swanson, Anush Moorthy, and JD Cock. 2018. VMAF: The journey continues. *Netflix Technology Blog* 25 (2018), 1.

[38] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 740–755.

[39] Nam Ling, C-C Jay Kuo, Gary J Sullivan, Dong Xu, Shan Liu, Hsueh-Ming Hang, Wen-Hsiao Peng, Jiaying Liu, et al. 2022. The Future of Video Coding. *APSIPA Transactions on Signal and Information Processing* 11, 1 (2022).

[40] Shu-yun Liu, Qun Hao, Yu-tong Zhang, Feng Gao, Hai-ping Song, Yu-tong Jiang, Ying-sheng Wang, Xiao-ying Cui, and Kun Gao. 2022. Single-image night haze removal based on color channel transfer and estimation of spatial variation in atmospheric light. *Defence Technology* (2022).

[41] Bin Lu, Lijuan Zhou, Shudong Zhang, Xin Chen, and Weidong Feng. 2021. Detection of small objects in complex long-distance scenes based on Yolov3. In *2021 4th International Conference on Data Science and Information Technology*. 98–102.

[42] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. 2019. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11006–11015.

[43] Hengyu Man, Chang Yu, Feng Xing, Yang Cheng, Bo Zheng, and Xiaopeng Fan. 2022. Deep learning-assisted video compression framework. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 3210–3214.

[44] Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Yue Yao, Liang Zheng, Pranamesh Chakraborty, Christian E. Lopez, Anuj Sharma, Qi Feng, Vitaly Ablavsky, and Stan Sclaroff. 2021. The 5th AI City Challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[45] Netflix. 2025. *VMAF - Video Multi-Method Assessment Fusion*. https://github.com/Netflix/vmaf/blob/master/libvmaf/README.md Accessed: 2025-02-19.

[46] The City of Bellevue. 2022. *Repository Name*. https://github.com/City-of-Bellevue/TrafficVideoDataset Accessed: 2024-08-15.

[47] NetMind Power. 2024. *NetMind Power: Rent GPUs at a Fraction of the Cost*. https://power.netmind.ai/gpuDashboard 2024-10-27.

[48] Graciela Ramirez-Alonso, Juan A Ramirez-Quintana, and Mario I Chacon-Murguia. 2017. Temporal weighted learning model for background estimation with an automatic re-initialization stage and adaptive parameters update. *Pattern Recognition Letters* 96 (2017), 34–44.

[49] Dillon Reis, Jordan Kupec, Jacqueline Hong, and Ahmad Daoudi. 2023. Real-Time Flying Object Detection with YOLOv8. *arXiv preprint arXiv:2305.09972* (2023).

[50] Adhi Rizal, Aries Suharso, Panji Abujabbar, and Munir Munir. 2020. Objective Quality Assessment of Multi-Resolution Video based on H. 264/AVC and H. 265/HEVC Encoding. In *Proceedings of the 7th Mathematics, Science, and Computer Science Education International Seminar, MSCEIS 2019, 12 October 2019, Bandung, West Java, Indonesia*.

[51] RunPod.io. 2024. *Deploy GPU Cloud*. https://www.runpod.io/console/deploy 2024-10-27.

[52] Jihoon Ryoo, Kiwon Yun, Dimitris Samaras, Samir R. Das, and Gregory Zelinsky. 2016. Design and evaluation of a foveated video streaming service for commodity client devices. In *2016 MMSys*. ACM, Article 6, 11 pages.

[53] Sepehr Sabour, Sanjeev Rao, and Majid Ghaderi. 2021. DeepFlow: Abnormal Traffic Flow Detection Using Siamese Networks. In *2021 IEEE International Smart Cities Conference (ISC2)*. 1–7. https://doi.org/10.1109/ISC253183.2021.9562915

[54] Avishek Saha, Young-Woon Lee, Young-Sup Hwang, Kostas E Psannis, and Byung-Gyu Kim. 2018. Context-aware block-based motion estimation algorithm for multimedia internet of things (IoT) platform. *Personal and Ubiquitous Computing* 22 (2018), 163–172.

[55] Parul Saini, Krishan Kumar, Shamal Kashid, Ashray Saini, and Alok Negi. 2023. Video summarization using deep learning techniques: a detailed analysis and investigation. *Artificial Intelligence Review* 56, 11 (2023), 12347–12385.

[56] Amanpreet Kaur Sandhu. 2021. Big data with cloud computing: Discussions and challenges. *Big Data Mining and Analytics* 5, 1 (2021), 32–40.

[57] Sandeep Singh Sengar and Susanta Mukhopadhyay. 2020. Motion segmentation-based surveillance video compression using adaptive particle swarm optimization. *Neural Computing and Applications* 32, 15 (2020), 11443–11457.

[58] Ajmal Shahbaz and Kang-Hyun Jo. 2020. Deep atrous spatial features-based supervised foreground detection algorithm for industrial surveillance systems. *IEEE Transactions on Industrial Informatics* 17, 7 (2020), 4818–4826.

[59] Zhenfeng Shao, Jiajun Cai, and Zhongyuan Wang. 2017. Smart monitoring cameras driven intelligent processing to big surveillance video data. *IEEE Transactions on Big Data* 4, 1 (2017), 105–116.

[60] Jude Tchaye-Kondi, Yanlong Zhai, Jun Shen, Dong Lu, and Liehuang Zhu. 2022. Smartfilter: an edge system for real-time application-guided video frames filtering. *IEEE Internet of Things Journal* 9, 23 (2022), 23772–23785.

[61] Vast.ai. 2024. *Global GPU Market: low-cost GPU rental*. https://vast.ai/ 2024-10-27.

[62] Shaohua Wan, Songtao Ding, and Chen Chen. 2022. Edge computing enabled video segmentation for real-time traffic monitoring in internet of vehicles. *Pattern Recognition* 121 (2022), 108146.

[63] Chen Wang, Aibek Musaev, Pezhman Sheinidashtegol, and Travis Atkison. 2019. Towards Detection of Abnormal Vehicle Behavior Using Traffic Cameras. In *Big Data – BigData 2019*, Keke Chen, Sangeetha Seshadri, and Liang-Jie Zhang (Eds.). Springer International Publishing, Cham, 125–136.

[64] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. 2022. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696* (2022).

[65] Shan Wang, Chuong Nguyen, Jiawei Liu, Kaihao Zhang, Wenhan Luo, Yanhao Zhang, Sundaram Muthu, Fahira Afzal Maken, and Hongdong Li. 2023. Homography Guided Temporal Fusion for Road Line and Marking Segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1075–1085.

[66] Yiding Wang, Weiyan Wang, Junxue Zhang, Junchen Jiang, and Kai Chen. 2019. Bridging the Edge-Cloud Barrier for Real-time Advanced Vision Analytics.. In *HotCloud*.

[67] Zhao Wang, Shiqi Wang, Xinfeng Zhang, Shanshe Wang, and Siwei Ma. 2019. Three-zone segmentation-based motion compensation for video compression. *IEEE Transactions on Image Processing* 28, 10 (2019), 5091–5104.

[68] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. 2020. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding* 193 (2020), 102907.

[69] Mathias Wien. 2003. Variable block-size transforms for H. 264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 7 (2003), 604–613.

[70] Nicolai Wojke and Alex Bewley. 2018. Deep Cosine Metric Learning for Person Re-identification. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 748–756. https://doi.org/10.1109/WACV.2018.00087

[71] Ping-Hao Wu, Ioannis Katsavounidis, Zhijun Lei, David Ronca, Hassene Tmar, Omran Abdelkafi, Colton Cheung, Foued Ben Amara, and Faouzi Kossentini. 2021. Towards much better SVT-AV1 quality-cycles tradeoffs for VOD applications. In *Applications of Digital Image Processing XLIV*, Vol. 11842. SPIE, 236–256.

[72] Yicheng Wu, Qiurui He, Tianfan Xue, Rahul Garg, Jiawen Chen, Ashok Veeraraghavan, and Jonathan T Barron. 2021. How to train neural networks for flare removal. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2239–2247.

[73] Xinggui Xu, Ping Yang, Hao Xian, and Yong Liu. 2019. Robust moving objects detection in long-distance imaging through turbulent medium. *Infrared Physics & Technology* 100 (2019), 87–98.

[74] Hyunho Yeo, Hwijoon Lim, Jaehong Kim, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2022. NeuroScaler: neural video enhancement at scale. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 795–811.

[75] Miao Zhang, Fangxin Wang, and Jiangchuan Liu. 2022. CASVA: Configuration-Adaptive Streaming for Live Video Analytics. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2168–2177.

[76] Qingyang Zhang, Hui Sun, Xiaopei Wu, and Hong Zhong. 2019. Edge video analytics for public safety: A review. *Proc. IEEE* 107, 8 (2019), 1675–1696.

[77] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. 2022. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. (2022).

[78] Anastasia V Zvezdakova, Dmitriy L Kulikov, Sergey V Zvezdakov, and Dmitriy S Vatolin. 2020. BSQ-rate: a new approach for video-codec performance comparison and drawbacks of current solutions. *Programming and computer software* 46 (2020), 183–194.

A ONLINE APPENDIX

A.1 Foreground extraction algorithms

Algorithm 4 outlines a detailed description of the object detection-based foreground extraction approach. The detection model is applied to a set of input videos $V = V_1, V_2, \ldots, V_N$, where each video undergoes several steps. For each video V_j, the algorithm retrieves essential properties as the frame rate (FPS), width (W), and height (H) to ensure the processed frames maintain the same structure and timing as the original. For each frame $F_{j,i}$ in video V_j, the YOLO model detects objects by identifying bounding boxes O_k for each object instance in the frame. This detection process captures only the regions of interest, effectively filtering out background pixels. Then, a binary mask $M_{j,i}$ of dimensions $H \times W$ is initialized for each frame. This mask is used to differentiate detected object regions (foreground) from the non-object regions (background). For each detected object, the corresponding pixels within the bounding box are marked as 1 in $M_{j,i}$. Using the binary mask, the algorithm modifies the frame to set all pixels outside detected regions to a green background color, represented as $(0, 255, 0)$ in RGB format. After processing each frame, the modified frame $F_{j,i}$ is written to a new video $V_{j,\text{fg}}$, which contains only the extracted foreground. Finally, once all frames in V_j are processed, the video writer is closed, and the resulting foreground-extracted video $V_{j,\text{fg}}$ is saved.

Instance segmentation-assisted foreground extraction approach is presented in Algorithm 5. Unlike the object detection method, which relies solely on bounding boxes to localize the detected objects, this approach uses instance segmentation masks. Each detected object is represented by a fine-grained segmentation mask, allowing for precise pixel-level separation of objects from the background. Non-maximum suppression (NMS) is applied to filter out overlapping detections, using confidence (T_{conf}) and IoU (T_{iou}) thresholds to retain only the most confident detections. A binary mask $M_{j,i}$ is created by adding the instance-specific segmentation masks M_k of selected object classes with confidence scores above a threshold (T_{conf}). The mask is then applied to isolate only the segmented object areas, making non-object areas more accurately

Algorithm 4 Object Detection-based DFE

Require: $V = \{V_1, V_2, \ldots, V_N\}$ ▷ Input videos
Require: $\mathbf{w}$ ▷ YOLO model weights
Ensure: $V_{\text{fg}} = \{V_{1,\text{fg}}, V_{2,\text{fg}}, \ldots, V_{N,\text{fg}}\}$ ▷ Foreground videos

Initialize YOLO model $\mathcal{M}_{\text{YOLO}}$ with $\mathbf{w}$
for each $V_j \in V$ **do**
  Retrieve video properties: FPS, W, H
  Initialize video writer for $V_{j,\text{fg}}$
  for each frame $F_{j,i} \in V_j$ **do**
    Detect objects: $\{O_k\} \leftarrow \mathcal{M}_{\text{YOLO}}(F_{j,i})$
    Initialize binary mask $M_{j,i}$
    for each bounding box O_k **do**
      $M_{j,i}[O_k] \leftarrow 1$ ▷ Mark detected regions
    end for
    Modify frame: $F_{j,i}[M_{j,i} = 0] \leftarrow (0, 255, 0)$
    Write modified frame to $V_{j,\text{fg}}$
  end for
  Encode $V_{j,\text{fg}}$ to H.264 format
  Close writer for $V_{j,\text{fg}}$
end for

Algorithm 5 Instance Segmentation-based DFE

Require: $V = \{V_1, V_2, \ldots, V_N\}$ ▷ Input videos
Require: $\mathcal{M}_{\text{YOLO}}, \mathbf{w}, \mathbf{hyp}$ ▷ model, weights, hyperparameters
Ensure: $V_{\text{seg}} = \{V_{1,\text{seg}}, V_{2,\text{seg}}, \ldots, V_{N,\text{seg}}\}$ ▷ Segmented videos

Initialize $\mathcal{M}_{\text{YOLO}}$ with $\mathbf{w}$ and load $\mathbf{hyp}$
for each $V_j \in V$ **do**
  Open V_j, get FPS, dimensions (W, H)
  Initialize video writer for $V_{j,\text{seg}}$
  while frame $F_{j,i}$ available **do**
    Resize $F_{j,i}$ to $(640, 640)$ and Convert $F_{j,i}$ to tensor
    Perform: $\{(O_k, M_k, C_k, p_k)\} \leftarrow \mathcal{M}_{\text{YOLO}}(F_{j,i})$
    Initialize mask $M_{j,i}$
    for each detection (O_k, M_k, C_k, p_k) **do**
      if $C_k \in$ selected classes and $p_k > T_{\text{conf}}$ **then**
        $M_{j,i} \leftarrow M_{j,i} + M_k$
      end if
    end for
    Set background green: $F_{j,i}[M_{j,i} = 0] \leftarrow (0, 255, 0)$
    Write $F_{j,i}$ to $V_{j,\text{seg}}$
  end while
  Close writer for $V_{j,\text{seg}}$
end for

distinguishable compared to simple bounding boxes. However, this method is computationally more intensive than the object detection model due to the pixel-level processing required for each segmentation mask.

Algorithm 6 presents the pseudo-code for a segmentation and trajectory prediction-based foreground extraction approach that integrates object tracking and trajectory prediction using a SORT (Simple Online and Realtime Tracking) tracker. By employing the SORT tracker, each detected object is assigned a unique ID, allowing the model to consistently associate objects across consecutive frames. The tracker updates its state with each new detection, and if an object reappears in later frames, it is reassigned its original ID. This ID-based tracking enables a continuous and stable trajectory for each object, promoting smoother transitions and frame-to-frame continuity.

For each frame, a binary mask $M_{j,i}$ is initialized with dimensions $(H \times W)$. The SORT tracker identifies objects with unique IDs, and for each tracked object $(O_k, M_k, C_k, \mathrm{ID}k, p_k)$ that meets the class and confidence threshold requirements, the corresponding segmentation mask M_k is added to Mj, i. This process isolates only the desired object regions, offering greater accuracy compared to frame-by-frame segmentation by maintaining the object's position and size, even when minor occlusions or fluctuations occur.

Algorithm 6 Segmentation and Trajectory Prediction-based Foreground Extraction

Require: $V = \{V_1, V_2, \ldots, V_N\}$ ▷ Set of input videos
Require: $\mathcal{M}_{\mathrm{YOLO}}$, **w**, **hyp** ▷ YOLO model with weights, hyperparameters
Ensure: $V_{\mathrm{sort}} = V_{1,\mathrm{sort}}, V_{2,\mathrm{sort}}, \ldots, V_{N,\mathrm{sort}}$ ▷ Tracked output videos

Initialize YOLO model $\mathcal{M}_{\mathrm{YOLO}}$ with **w**, load **hyp**
Set device $\mathcal{D}$ and precision (half if $\mathcal{D}$ = GPU)
Define output folder
for each $V_j \in V$ **do**
    Open V_j, retrieve FPS FPSj and dimensions (W, H)
    Initialize video writer for Vj, sort, initialize SORT tracker $\mathcal{T}_{\mathrm{SORT}}$
    for each frame $F_{j,i}$ from V_j **do**
        1. $F_{j,i} \leftarrow \mathrm{letterbox}(F_{j,i}, 640)$ ▷ Resize to $(640, 640)$
        2. $F_{j,i} \leftarrow \mathrm{Tensor}(F_{j,i})$, move to $\mathcal{D}$
        3. $\{(O_k, M_k, C_k, p_k)\} \leftarrow \mathrm{NMS}(\mathcal{M}_{\mathrm{YOLO}}(F_{j,i}), T_{\mathrm{conf}}, T_{\mathrm{iou}})$
        4. Update SORT tracker: $\{(O_k, \mathrm{ID}_k)\} \leftarrow \mathcal{T}_{\mathrm{SORT}}.update(\{O_k, M_k\})$
        5. Initialize mask $M_{j,i} \in \{0, 1\}^{H \times W}$
        for each tracked $(O_k, M_k, C_k, \mathrm{ID}_k, p_k)$ **do**
            if $C_k \in$ selected classes and $p_k > T_{\mathrm{conf}}$ **then**
                1. $M_{j,i} \leftarrow M_{j,i} + M_k$ ▷ Update mask with object region
                2. Annotate $F_{j,i}$ with ID_k
            end if
        end for
        6. $F_{j,i}[M_{j,i} = 0] \leftarrow (0, 255, 0)$ ▷ Set background to green
        7. Write $F_{j,i}$ to $V_{j,\mathrm{sort}}$
    end for
    Close writer for $V_{j,\mathrm{sort}}$
end for

A.2 Dataset diversity

(1) **Camera shivering** can adversely affect traffic footage quality. We identified and chose 5 out of 145 videos with significant instability, emphasizing the need to address this issue for accurate traffic monitoring and analysis.

(2) **Night time videos**, categorized into two classes based on darkness levels. The first set includes 5 profoundly dark videos, with only vehicle lights visible, while the second set consists of 5 videos taken during the early night and pre-dawn, with additional ambient illumination from sources like moonlight and streetlights.
(3) **Camera locations** are at major intersections or along lengthy streets. Cameras at intersections capture more slow-moving objects than those in suburban areas. We considered 5 videos from major intersections and 5 videos from straight streets to account for these differences.
(4) **Traffic volume** is assessed by counting moving objects in video footage. Videos with more than 15 objects are classified as high traffic, while those with fewer are considered low traffic. Our evaluation used 5 videos each for low and high traffic volume conditions.
(5) **Weather conditions** like rain, snow, fog, high temperature, and wind can affect traffic camera quality and reliability. To evaluate system performance, 5 videos captured under various weather conditions were used, comparing the baseline schemes and *CLOUD-CODEC*.
(6) **Severe conditions** in traffic videos refer to situations with impeded traffic flow due to factors like high traffic volume, nighttime, inclement weather, or camera instability. Five videos from the dataset, representing traffic under severe conditions, were used for system evaluation.
(7) **Pedestrian activity** refers to the presence, movement, and behavior of pedestrians in various traffic scenarios, including crowded sidewalks, crosswalk usage, biking, and interactions with vehicular traffic. Five videos from the AICC21 dataset are considered in this scenario.

A.3 H.264 and AV 1 video encoder implementation parameters

The default parameters of the H.264 encoding are as follows:

- `Constant Rate Factor (CRF)` - The default value is `23`. The range of the CRF scale is `0–51`, where `0` is lossless, and `51` is the worst quality possible. A lower value generally leads to higher quality.
- `Preset`-The default setting is `medium`. This setting is used to balance video quality and encoding speed. A `slower` preset achieve better quality, while `very fast` or `faster` presets provide high encoding speed but at the expense of video quality.
- `Profile`-The default setting is `high`, which allows for advanced encoding methods and a better compression ratio. Other profiles, like `baseline` and `main` are primarily used with very old and obsolete devices.

FFmpeg libaom-av1 library is used to implement the *AV1 video encoder* with the following parameters:

- `Constant Rate Factor (CRF)`-The CRF value can be from 0 to 63 in `AV1`. Lower values provide better quality but also larger file size. A CRF value of 31 in AV1 produces a quality level approximately equivalent to a CRF value of 23 in H.264, a level that is generally perceived as visually equal [71]. Therefore, 31 has been chosen as the CRF parameter.
- `CPU Used`-This parameter balances the efficiency between video quality and encoding speed, with values ranging from 0 to 8. Lower values result in slower encoding speeds but higher video quality, and vice-versa. Since in H.264 we have chosen default `medium preset`, in AV1 we used value 4 for `CPU Used`.
- `Row Multi-threading` allows row-based multi-threading to enhance the encoding process's speed with minimal impact on quality. Using `-row-mt 1` activates this feature, while `-row-mt 0` deactivates it. In our work, we have opted to enable this feature by setting the parameters to 1.

Table 3. Statistical summary of video size reduction rate (%) for *CLOUD-CODEC* and competing approaches across four different datasets.

Dataset	Error	AV1	CloudSeg	Detection	Segmentation	Segm& Sort	CLOUD-CODEC
AICC21	**Mean**	37.9	49.9	71.9	69.3	64.2	**74.9**
	Std Dev	11.22	13.80	9.82	10.48	11.57	9.94
	CI Lower	34.83	45.14	69.97	67.25	61.90	70.94
	CI Upper	38.15	52.65	73.86	71.41	66.49	76.89
Bellevue	**Mean**	22.5	64.13	78.9	74.5	70.2	**80.3**
	Std Dev	9.00	12.91	10.26	10.77	12.48	9.02
	CI Lower	19.11	52.90	66.57	62.79	57.29	72.93
	CI Upper	31.99	71.36	81.24	78.19	75.14	83.84
DETRAC	**Mean**	39.5	55.2	76.6	72.5	69.62	**78.8**
	Std Dev	10.61	8.65	9.26	10.24	11.23	9.00
	CI Lower	35.10	50.57	65.89	64.83	62.52	69.43
	CI Upper	47.03	64.07	78.62	74.41	72.04	80.62
AAU	**Mean**	39.97	64.49	80.25	76.24	73.94	**82.40**
	Std Dev	9.39	6.63	13.38	17.65	18.12	12.78
	CI Lower	34.68	61.9	72.84	66.47	63.91	75.32
	CI Upper	47.56	67.06	87.66	86.01	83.98	89.48

A.4 Video size reduction rate results

A.5 GPU cost analyses

For storing original videos, we selected Amazon EBS (Elastic Block Store) HDD cloud service and the corresponding pricing details are summarized in Table 4. Amazon EBS offers two types of HDD services: Cold HDD and Throughput-Optimized HDD. Cold HDD (sc1) is ideal for infrequent, archival storage due to its cost-effectiveness, while Throughput-Optimized HDD (st1) is suited for large, sequential workloads like video streaming and logging, prioritizing sustained throughput. Therefore, we focus on the pricing of the TO HDD (st1) and this service is billed per GB for a month but charged in per-second increments [4] as presented in Eq 3.

$$\text{Cost} = P \times S \times \frac{T_u}{T_m} \tag{3}$$

Where P represents the price per GB-month, such as 0.045 USD/GB-month. S denotes the provisioned storage in gigabytes (GB). T_u is the usage duration measured in seconds, while T_m refers to the total number of seconds in a month.

For GPU processing pricing, we analyze the costs of H100 and A100 GPU units offered by four major service providers (Table 4): Amazon EC2 [5], Vast.AI [61], RunPod [51], HPC-AI [30], and NetMind Power [47]. Each provider offers GPU rental services, and we consider single A100 and H100 GPU units for a straightforward price comparison. Among these providers, NetMind Power offers the lowest hourly rate at $1.04 for the A100 and $2.00 for the H100. Consequently, our cost analysis focuses on the pricing rates offered by NetMind Power. *CLOUD-CODEC* achieves average encoding rates of approximately 575 frames per second (fps) on the H100 GPU and 215 fps on the A100 GPU.

Below are key points that strengthen this claim:

(1) **Future Decline in GPU Prices**: GPU costs, especially in cloud environments, are expected to decline over time due to advancements in technology, increased competition, and economies of scale. This trend will improve the long-term economic feasibility of *CLOUD-CODEC* encoding.

Table 4. Overview of HDD and GPU Service Providers and Their Usage Costs

HDD Provider ($/GB/mon)	AWS EBS	**Cold** 0.015	**Optimized** 0.045
GPU Provider ($/hours)		**A100**	**H100**
	AWS EC2 [5]	1.84	4.91
	Vast.AI [61]	1.06	2.85
	RunPod [51]	1.19	2.79
	HPC-AI [30]	1.25	2.5
	NetMind Power [47]	1.04	2.0

Table 5. Video Size and GPU Processing Time

Storing Scenarios	**Datasets**	**Video Size (GB)**		**CLOUD-CODEC Proc. Time (h)**	
		Original	Encoded	A100	H100
Static	AICC21	4.9	1.3	0.47	0.17
	Bellevue	62.3	12.6	14.03	5.27
Incremental	Synthetic	1000	210	230	84.6

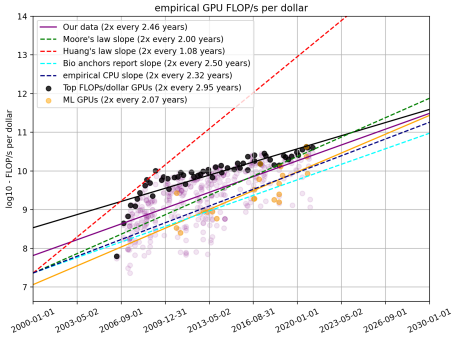


Fig. 12. Trends in GPU price-performance improvement over time, measured as floating-point operations per second per dollar (FLOP/s per $) on a logarithmic scale. [29]. Note: The "Our data" label in the figure refers to Epoch AI.

(2) **Ownership of In-House GPU Resources**: CCTV data centers with in-house GPU infrastructure can avoid the recurring costs of cloud providers like AWS. For these centers, operational expenses for encoding, such as electricity, maintenance, and cooling, are generally lower than cloud GPU costs, making in-house GPUs a cost-saving alternative that leverages existing resources.

(3) **Emerging GPU Rental Models and Competitive Pricing**: Certain cloud providers offer cost-effective GPU rental models, like spot instances and reserved pricing, which can significantly reduce GPU costs. As competition increases, rental rates are becoming more affordable, with providers like Vast.AI and HPC-AI offering lower-cost alternatives to AWS EC2, making GPU-based encoding more accessible and economical.

(4) **Higher Price-Performance Improvements in GPUs**: Recent analyses [29] published by Epoch AI on GPU price-performance trends strongly support the potential long-term cost-effectiveness of GPU-based encoding and storage solutions. By examining data from 470 GPU models released between 2006 and 2021, the authors report that FLOP/s per dollar approximately doubles every 2.5 years. This rate of improvement is notable, especially compared to Moore's law, which doubles every two years, highlighting significant gains in the affordability of GPU processing over time, as shown in Figure 12.

Received 9 June 2024; revised 27 February 2025; accepted 31 May 2025