ST-360: Spatial–Temporal Filtering-Based Low-Latency 360-Degree Video Analytics Framework

JIAXI LI, University of Illinois Urbana-Champaign, USA
JINGWEI LIAO, George Mason University, USA
BO CHEN, University of Illinois Urbana-Champaign, USA
ANH NGUYEN, George Mason University, USA
ADITI TIWARI, University of Illinois Urbana-Champaign, USA
QIAN ZHOU, City University of Hong Kong, China
ZHISHENG YAN, George Mason University, USA
KLARA NAHRSTEDT, University of Illinois Urbana-Champaign, USA

Recent advances in computer vision algorithms and video streaming technologies have facilitated the development of edge-server-based video analytics systems, enabling them to process sophisticated real-world tasks, such as traffic surveillance and workspace monitoring. Meanwhile, due to their omnidirectional recording capability, 360-degree cameras have been proposed to replace traditional cameras in video analytics systems to offer enhanced situational awareness. Yet, we found that providing an efficient 360-degree video analytics framework is a non-trivial task. Due to the higher resolution and geometric distortion in 360-degree videos, existing video analytics pipelines fail to meet the performance requirements for end-to-end latency and query accuracy. To address these challenges, we introduce the innovative ST-360 framework specifically designed for 360-degree video analytics. This framework features a spatial-temporal filtering algorithm that optimizes both data transmission and computational workloads. Evaluation of the ST-360 framework on a unique dataset of 360-degree first-responders videos reveals that it yields accurate query results with a 50% reduction in end-to-end latency compared to state-of-the-art methods.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Distributed algorithms**.

Additional Key Words and Phrases: 360-Degree Video Analysis, Edge Computing, Smart Filtering

1 INTRODUCTION

Recent advancements in Deep Neural Network (DNN) enable the possibility of complicated vision tasks such as object detection. These DNN-based object detectors, when integrated with live video streaming services, form the video analytics systems that are able to process sophisticated real-world tasks such as traffic surveillance and workspace monitoring. The state-of-the-art video analytics frameworks[25, 26, 28, 34, 35, 45, 45, 59, 62, 63, 79] utilize an edge-server offloading modality (Figure 1), where the remote edge device, such as a camera, deployed at locations like crossroads or factories, continuously captures and streams video frames to the server. This server

Authors' addresses: Jiaxi Li, jiaxili3@illinois.edu, University of Illinois Urbana-Champaign, Urbana, Illinois, USA; Jingwei Liao, George Mason University, Fairfax, USA, jliao2@gmu.edu; Bo Chen, University of Illinois Urbana-Champaign, Urbana, USA, boc2@illinois.edu; Anh Nguyen, George Mason University, Fairfax, USA, anguy59@gmu.edu; Aditi Tiwari, University of Illinois Urbana-Champaign, Urbana, USA, aditit5@illinois.edu; Qian Zhou, City University of Hong Kong, Hong Kong, China, qian.zhou@cityu.edu.hk; Zhisheng Yan, George Mason University, Fairfax, USA, zyan4@gmu.edu; Klara Nahrstedt, University of Illinois Urbana-Champaign, Urbana, USA, klara@illinois.edu.

ACM 1551-6865/2024/9-ART
https://doi.org/10.1145/3694685

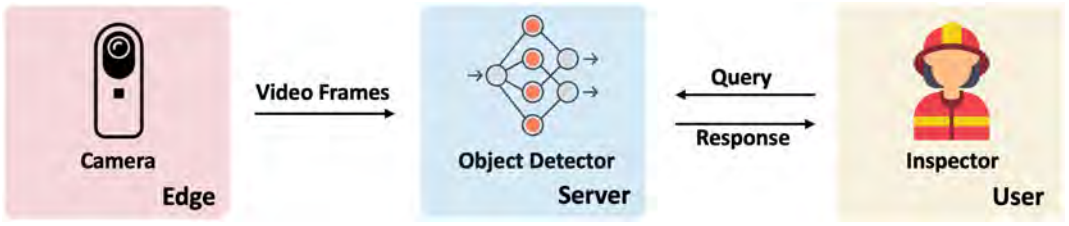

Fig. 1. High level abstraction of the edge-server offloading modality for video analytics framework.

then constantly performs object detection on the received frames. Compared to the server, the edge is lighter and less powerful in hardware, hence the computationally intensive object detection workloads are generally offloaded to the server[25, 28, 35, 45, 59, 62, 63]. Then, based on the detection results, the server responds to queries from the human inspector. Examples of these queries include counting the number of vehicles per hour at a crossroad in traffic surveillance or verifying that all workers are wearing helmets in workspace monitoring scenarios. Query accuracy evaluates if the analytics system correctly responds to user queries, which is closely tied to the object detection accuracy.

Commercial cameras used in current video analytics frameworks typically record a limited field of view, usually less than 120 degrees[22, 28]. For tasks where omnidirectional situational awareness is important (e.g., crossroad traffic monitoring), multiple cameras need to be purchased and installed, which leads to extra material and labor costs. One of the solutions is to replace traditional cameras with panoramic cameras such as 360-degree cameras[10, 12, 27, 48, 65]. However, 360-degree video analytics is a non-trivial task due to the larger frame size and geometric distortion. The increased frame size results in significantly longer latencies during network transmission and object detection stages. Meanwhile, geometric distortion leads to misidentified or missed objects, thus reducing detection accuracy.

To be specific, 360-degree videos are generally of 4K resolution. Transmitting video frames of such size results in increased transmission and propagation delays, hence leading to greater end-to-end latency, particularly under dynamic and unstable network conditions between the edge and the server. To improve network transmission, previous works[6, 42, 52, 56, 64] have proposed a tile-based adaptive streaming pipeline for 360-degree video streaming, which segments each 360-degree video frame into sub-frames, or tiles, and transmits each tile with different bitrates to reduce latency. However, most of those works are optimized for better video watching experience rather than higher vision query accuracy, or assume on-demand streaming[13, 20, 71], and therefore cannot be directly applied to our case of live video analytics. Moreover, in the object detection stage, the increased number of pixels in 360-degree frames extends detection times and increases GPU memory usage, thereby causing longer end-to-end latency and higher hardware requirements[27, 71].

On the other hand, object detection on 360-degree videos suffers from reduced accuracy due to geometric distortion[9, 21, 27, 54, 66, 71]. Commercial 360-degree cameras (e.g., RICOH THETA series) often store 360-degree videos in equirectangular projection (ERP) format[70, 75], which projects 360-degree videos into rectangular-shaped videos, leading to geometric distortion at the high latitude areas (north pole and south pole areas)[2, 27]. Geometric distortion can alter the appearance of objects, causing the traditional 2D object detectors[14, 15, 46, 47] to either miss the object or incorrectly classify the object into the wrong category, thereby reducing detection accuracy. Prior research has introduced a dual-projection method[27, 44, 65] that segments each ERP-formatted video frame into 6 ERP Tiles. Each of these ERP Tiles is then projected onto a CMP Tile, effectively reducing distortion. However, incorporating the dual-projection process also increases system latency due to the additional computational overhead.

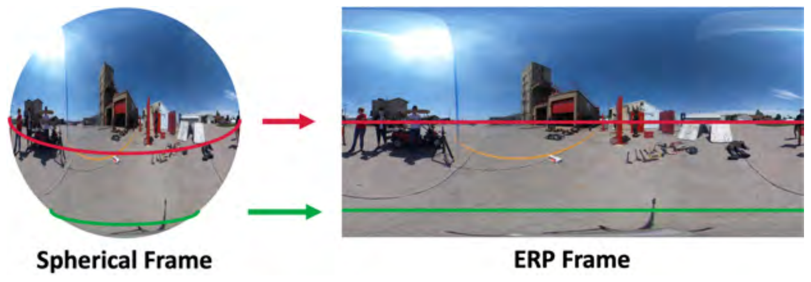

Fig. 2. Spherical video frame projected to ERP format. The green line and red line in the left spherical frame are of different circumferences, and are projected to lines with a uniform width in the right ERP frame. Note that south pole areas (green line) are more distorted than equatorial areas (red line) on the right frame.

In this paper, we show that existing 2D video analytics pipelines fail to meet the latency and accuracy requirements when directly applied to the 360-degree case. Therefore, to tackle the above-mentioned challenges, we propose ST-360, the first online video analytics framework customized for 360-degree videos. The ST-360 framework integrates a tile-based dual-projection approach in the analytics pipeline for enhanced accuracy performance. To reduce latency, the ST-360 framework features the novel spatial filtering and temporal filtering modules that can be efficiently executed at the edge device. The spatial filtering module removes unimportant areas inside each ERP Tile through low-level image feature extraction. The temporal filtering module filters out repeating ERP Tiles by comparing low-level image similarity. Considerable latency reduction on network transmission and object detection stages is achieved by the ST-360 framework compared to the state-of-the-art methods. Evaluation results from a test dataset comprising 14 360-degree videos indicate that the ST-360 framework achieves an average reduction of 50% in end-to-end latency while delivering accurate query results.

The remainder of this paper is structured as follows. Section 2 presents an overview introduction to 360-degree videos and video analytics requirements, assesses the latency and accuracy performance of applying 2D video analytics systems on 360-degree videos, and examines the accuracy enhancements achieved through the dual-projection process. Section 3 introduces the ST-360 framework with a description of the system components along with the data and control pipelines. Section 4 and Section 5 describe the motivation of the temporal and spatial filtering modules, evaluate their feasibility on an edge-server architecture, and provide a detailed explanation on the implementation of these filtering algorithms. The end-to-end evaluation of the ST-360 framework is shown in Section 6. Section 7 discusses the related work and Section 8 concludes the paper.

2 BACKGROUND AND PRELIMINARY STUDY

2.1 360-Degree Cameras and Videos

360-degree cameras, typically consisting of multiple ultra-wide fisheye lenses facing different directions, create spherical frames by stitching together images captured from each individual lens. For better compatibility in data transmission and storage, most commercial 360-degree cameras, such as the RICOH THETA series, record videos in the equirectangular projection (ERP) format. The ERP format projects spherical video frames onto normal 2D video frames (Figure 2). During the projection process, geometric distortion arises from the fact that circumferences of spherical frames in high-latitude (north pole and south pole) areas are shorter than these in low-latitude areas. However, the ERP format projects the frame content of different circumferences onto a 2D frame with a uniform width, thereby introducing additional pixels into the projected frame. Distortion is created due to the shift in the relative geometric positions among the pixels after incorporating these extraneous pixels. Previous studies[18, 27, 54, 60, 73] indicate that these distortions can significantly reduce accuracy in video analytics tasks such as object bounding box detection.

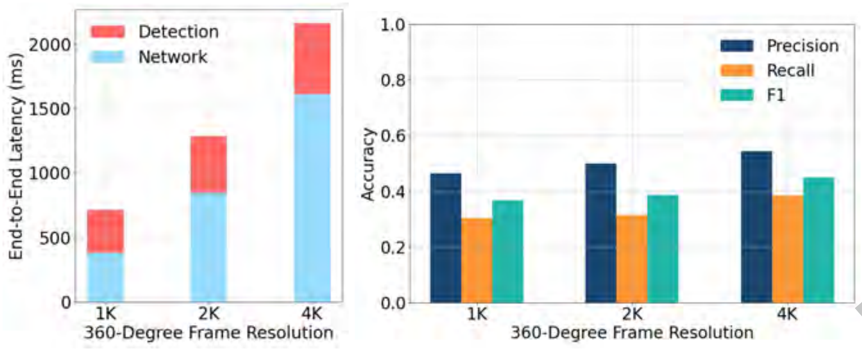

Fig. 3. End-to-end latency and object bounding box detection accuracy results on ERP-formatted 360-degree videos using existing 2D video analytics methods. Object detection is performed using a YOLOv3 object detector[46] pre-trained on the COCO dataset[30].

2.2 Video Analytics Requirements

Existing works for 2D video analytics have targeted at an end-to-end latency of under 1000 milliseconds[8, 17, 29, 37, 59, 63, 69, 72, 74, 78] and an evaluation accuracy of around 0.7 to 0.9 measured in mAP (mean Average Precision) or F1 score[1, 17, 37, 59, 61, 77]. We conducted a preliminary experiment to test if those requirements can be satisfied by applying the state-of-the-art 2D video analytics methods on the 360-degree case. Specifically, we built an edge-server architecture with a laptop of a 4-core CPU as the edge and a desktop of a 6-core CPU and a NVIDIA GeForce RTX 3080 Ti graphics (12GB memory) as the server. We selected a dataset of 6 ERP-formatted 360-degree videos resized to different frame resolutions to simulate the camera stream. The video frames were compressed under the JPEG format at the edge device before sending to the server. A YOLOv3[46] object detector pre-trained on the COCO dataset[30] was running at the server to detect objects and answer queries. We selected the human as the object of interest and evaluated the performance of the system in answering object bounding box detection query. The detection results for each frame were compared to the ground-truth object bounding box coordinates that were manually labeled.

Figure 3 shows the end-to-end latency and object bounding box detection accuracy results of this preliminary experiment. We found an increasing F1 socres as frame resolution increases. However, none of the F1 scores was within the desirable range. We noticed that none of the Recall accuracies was above 40%, indicating that a certain number of actual objects were missed or incorrectly classified by the object detector. In addition, among all resolutions, only the 1K resolution (lowest accuracy) satisfied the above mentioned requirements for 2D video analytics tasks. Furthermore, it is noticeable that in comparison to the object detection stage, the network transmission stage accounts for the majority of the end-to-end latency at higher resolutions (2K and 4K).

The preliminary experiment results indicate that existing 2D video analytics pipelines, when directly applied to the 360-degree case, do not satisfy the desired query accuracy requirement. Furthermore, they do not fulfill the end-to-end latency requirement at 2K and 4K resolutions.

2.3 Dual-Projection

Previous studies have identified geometric distortion as a primary factor contributing to the detection accuracy degradation when 2D object detectors are directly applied to 360-degree videos[9, 54, 60, 68, 70, 73]. We also notice severe geometric distortion in high-latitude areas of the ERP-formatted 360-degree videos in our dataset. For example, the left image in Figure 4 represents an example of ERP-formatted 360-degree frames. As indicated

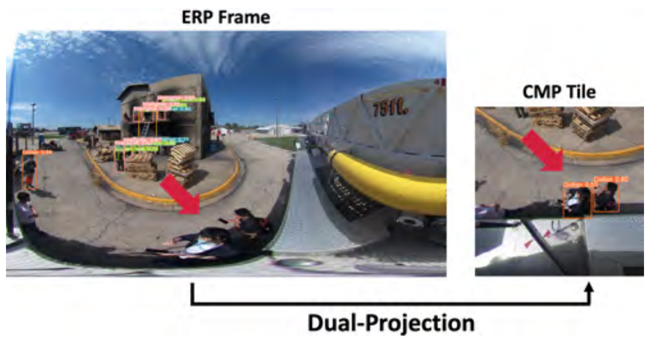

Fig. 4. An example showing the distortion in ERP-formatted 360-degree frames. The distortion is reduced after dual-projecting the ERP-formatted frame to CMP Tile.

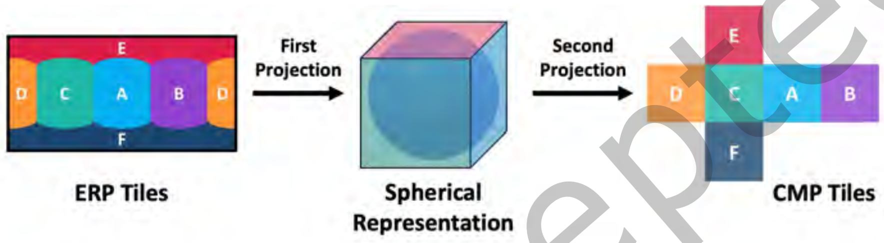

Fig. 5. The dual-projection process. The ERP Frame is segmented into 6 ERP Tiles, which are transformed into the spherical representation under the first projection. These spherical tiles are transformed into 6 CMP Tiles using the cube mapping method in the second projection.

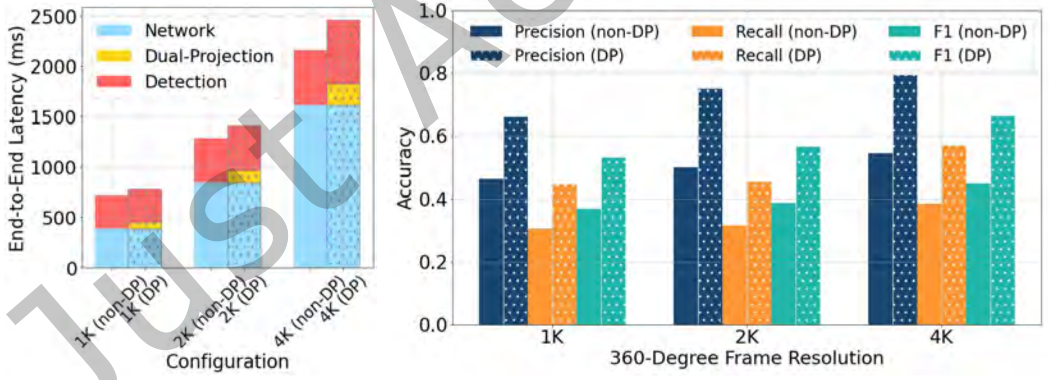

Fig. 6. End-to-end latency and object bounding box detection query accuracy achieved on 360-degree videos using 2D video analytics methods with and without dual-projection.

by the arrow, the two humans in the south pole areas are distorted and are missed by the detector. The missed detected objects potentially explain the low Recall accuracy discussed in Section 2.2.

One approach to remove distortion in ERP-formatted frames is through a tile-based dual-projection (DP) process[27, 44, 65]. The key idea is to transform each ERP-formatted frame to the spherical representation, and apply it with the cube mapping method[58]. Figure 5 provides an overall description of this process. Specifically,

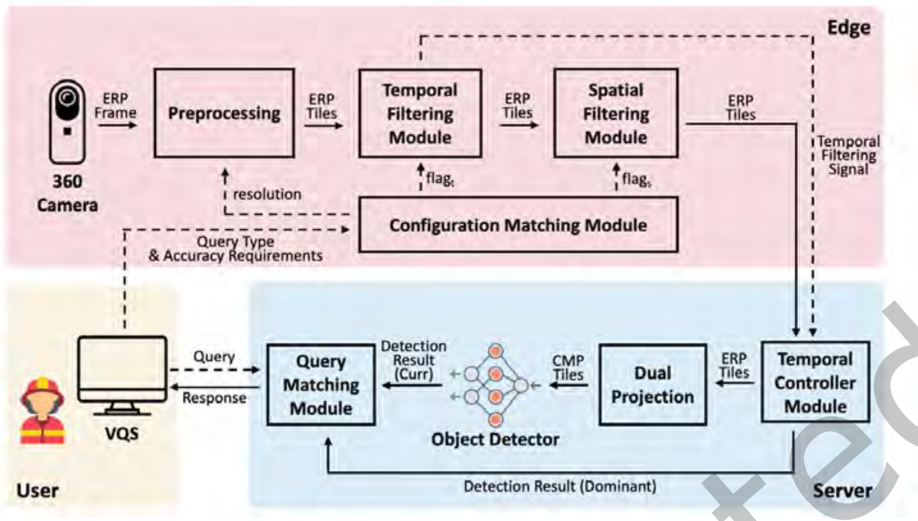

Fig. 7. Overview of the ST-360 framework.

each ERP-formatted frame is segmented into 6 ERP Tiles, which are transformed to the spherical representation under the first projection. Then, these spherical tiles are transformed to the corresponding CMP Tiles under the second projection (cube mapping). The detection after the dual-projection is performed on the 6 CMP Tiles, which involve less distortion. For example, the right image of Figure 4 represents a CMP Tile projected from an ERP-formatted frame. The dual-projection process reduces the distortion on the south pole areas, which enables the object detector to discover the object of interest (human).

To investigate the accuracy improvement of incorporating the dual-projection process in 360-degree video analytics systems, we repeated the preliminary experiment for the dual-projection case, and compared the results to those of the non-dual-projection case (results achieved in Section 2.2). The comparison of end-to-end latency and query accuracy results are presented in Figure 6. In particular, we found significant accuracy improvements achieved in the dual-projection case, with an average of 23% improvement in Precision, 16% improvement in Recall, and 19% improvement in F1 score among all resolutions. Among all the configurations, the dual-projection case at 4K resolution achieved the best accuracy with a 66.5% in F1 score, which is close to the satisfactory range discussed in Section 2.2.

While the dual-projection approach greatly enhances query accuracy, it also adds more processing time, averaging a 10% increase in end-to-end latency across all resolutions due to the projection computation. The end-to-end latency for higher resolutions (2K and 4K) remains significantly above the desired range.

3 ST-360 FRAMEWORK OVERVIEW

To address the high latency, we introduce the ST-360 framework, an edge-server-based solution for online 360-degree video analytics. In addition to the dual-projection process, this framework integrates the novel temporal and spatial filtering modules at the edge. The temporal filtering module computes low-level image similarity between ERP Tiles, and bypasses the network transmission and object detection processes of the repetitive ERP Tile if high similarity is found (details in Section 4). The spatial filtering module locates and compresses areas in each ERP Tile lacking objects of interest to speed up the network transmission and object detection processes (details in Section 5).

The overview of the ST-360 framework is presented in Figure 7, where the solid lines represent the data flow and the dotted lines represent the control flow. The ST-360 framework involves the edge, server, and user components.

3.1 User

The user component features a web-based Viewing and Query Service (VQS), which allows the user to access and view the query responses. The following query types are supported by the ST-360 framework, including 1) object counting query, which returns the amount of a given category of object detected in the camera, and 2) object bounding box detection query, which returns the bounding box coordinates for every occurrence of a given category of object detected in the camera. Additionally, the user component enables the user to modify the query type and set specific accuracy requirements. For the object counting query, users can define their desired Accuracy Rate (explanation of this metric in Section 6.1.5), while for the object bounding box detection query, they have the option to specify one or several accuracy requirements, including Precision, Recall, and F1 score.

3.2 Edge

The edge component consists of a 360-degree camera supported with limited computation power. It is responsible for determining the functional configuration, capturing the raw ERP Frame, resizing and segmenting the ERP Frame to ERP Tiles, compressing the ERP Tiles based on the temporal and spatial filtering modules, and transmitting the compressed ERP Tiles to the server component. The detailed pipeline of the edge component is presented in Algorithm 1.

Prior to initiating the online video analytics pipeline, the configuration matching module determines the functional configuration of the current pipeline based on the user-specified query type and accuracy requirements. The functional configuration of our ST-360 framework comprises two key elements: the schema and the resolution. The schema options are as follows: 1) FULL, where neither temporal nor spatial filtering modules is active; 2) TF, activating only the temporal filtering module; 3) SF, activating only the spatial filtering module; and 4) TFSF, where both temporal and spatial filtering modules are active. As for resolution, the choices available are 1K, 2K, and 4K. The selection of the functional configuration is based on offline pre-evaluated results from running all possible configurations on a dataset of 14 one-minute 360-degree videos (detailed in Section 6). For every query type, the functional configuration that satisfies all the specified accuracy requirements with the minimal end-to-end latency is selected according to Table 5 and 6. Depending on the selected functional configuration, the configuration matching module outputs the resolution variable, $resolution$, as well as two boolean variables, $flag_t$ and $flag_s$, which respectively indicate whether the temporal and spatial filtering modules are activated (Step 1-1, line 1).

After the online pipeline starts, each captured ERP Frame undergoes preprocessing, which includes resizing to the chosen resolution in Step 1-2, followed by segmentation into 6 ERP Tiles in Step 1-3. In Steps 1-4, the temporal filtering signal array ($TR[1...6]$) is initialized as an array of boolean values. A True value indicates that the dual-projection and object detection processes are required in the server component for a particular ERP Tile, vice versa. Depending on the $flag_t$, the temporal filtering module is executed in Step 1-5 to remove repetitive ERP Tiles and update the temporal filtering signal array (details in Section 4.4). Similarly, depending on the $flag_s$, the spatial filtering module is executed in Step 1-6 to compress the ERP Tiles (details in Section 5.3). The result ERP Tiles are further compressed under the JPEG format during Step 1-7, and are sent to the server component along with the temporal filtering signal array in Step 1-8.

The primary computational bottleneck of Algorithm 1 arises from the temporal and spatial filtering steps. Consequently, the complexity of Algorithm 1 depends on $flag_t$ and $flag_s$. If either of these boolean variables is True, the overall complexity can be summarized as $O(N \times W \times H)$ where N represents the number of ERP Tiles captured and $W \times H$ denotes the size of each ERP Tile. In this case, each EPR Tile is subject to temporal and/or spatial filtering, each with a complexity of $O(W \times H)$. The complexity analysis of the temporal and spatial filtering modules are described in Section 4.4 and 5.3. If both $flag_t$ and $flag_s$ are False, the overall complexity reduces to $O(N)$.

3.3 Server

The server component consists of a device equipped with powerful CPUs and GPUs. The server component is responsible for dual-projecting and detecting the received ERP Tiles, and answering user-specified queries based on the detection results. The detailed pipeline of the server component is presented in Algorithm 2.

The server component utilizes the array *DominantDR*, which stores the object detection results of the reference ERP Tiles maintained in the temporal filtering module (detailed in Section 4.4). These detection results are employed when the currently received ERP Tiles exhibit high similarity to the previously received reference ERP Tiles and are consequently filtered out.

After the online pipeline starts, the temporal controller module within the server component receives the ERP Tile array along with the corresponding temporal filtering signal array in Step 2-1 (line 2). For ERP Tiles labeled True in the signal array, they are dual-projected into CMP Tiles (Step 2-2), followed by 2D object detection (Step 2-3) performed with a YOLOv3 object detector[46] that was trained offline on the COCO dataset[30]. The corresponding entry in *DominantDR* is updated to store the latest detection results (Step 2-4), which are later forwarded to the query matching module. Conversely, ERP Tiles labeled False in the signal array bypass both the dual-projection and object detection processes. The corresponding results in *DominantDR* are utilized and sent to the query matching module (Step 2-5). Based on the user-specified query, the query matching module processes the raw detection results, converting them into user-friendly query responses (Step 2-6) that are then transmitted to the user-side VQS (Step 2-7).

The complexity of Algorithm 2 depends on N', the number of ERP Tiles after temporal filtering. Each unfiltered ERP Tile undergoes the dual-projection and object detection processes, with complexities of $O(W \times H)$ and $O(W \times H \times D)$ respectively, where $W \times H$ represents the size of each ERP Tile and D denotes the depth of the convolutional neural network used by the object detection algorithm. Therefore, the overall complexity can be simplified to $O(N' \times W \times H \times D)$.

Algorithm 1 ST-360 Edge Pipeline

```
Input:  $type$: The user-specified query type
        $accuracy$: The user-specified accuracy requirements
 1: $resolution, flag_t, flag_s \leftarrow ConfigurationMatching(type, accuracy)$      ▹ Step 1-1
 2: while True do
 3:     $frame \leftarrow Capture()$
 4:     $frame \leftarrow Resize(frame, resolution)$                                  ▹ Step 1-2
 5:     $ET[1...6] \leftarrow SegmentToERP(frame)$                                   ▹ Step 1-3
 6:     $TFSignal[1...6] \leftarrow [True] * 6$                                      ▹ Step 1-4
 7:     if $flag_t == True$ then
 8:         $ET[1...6], TFSignal[1...6] \leftarrow TemporalFiltering(ET[1...6])$     ▹ Step 1-5
 9:     if $flag_s == True$ then
10:         $ET[1...6] \leftarrow SpatialFiltering(ET[1...6])$                       ▹ Step 1-6
11:     $ET[1...6] \leftarrow JPEG(ET[1...6])$                                       ▹ Step 1-7
12:     $Send(ET[1...6], TFSignal[1...6])$                                          ▹ Step 1-8
```

4 TEMPORAL FILTERING

In the next two sections, we provide a comprehensive introduction to the spatial and temporal filtering modules, discussing the motivation, feasibility study, and specific pipeline of the filtering algorithms.

Algorithm 2 ST-360 Server Pipeline

```
Input: $query$: The user-specified query
Initialize: $CurrDR[1...6]$: The detection results for the ERP Tiles of the current timestamp
            $DominantDR[1...6]$: The detection results for the reference ERP Tiles
 1: while True do
 2:     $ET[1...6], TFSignal[1...6] = Receive()$                          ▷ Step 2-1
 3:     for $i = 1, i++, i \leq 6$ do                    ▷ Iterating through the 6 ERP Tiles
 4:         if $TFSignal[i] == True$ then
 5:             $CMPTile_i \leftarrow DualProjection(ET[i])$              ▷ Step 2-2
 6:             $CurrDR[i] \leftarrow 2DObjectDetection(CMPTile_i)$       ▷ Step 2-3
 7:             $DominantDR[i] \leftarrow CurrDR[i]$                      ▷ Step 2-4
 8:         else if $TFSignal[i] == False$ then
 9:             $CurrDR[i] \leftarrow DominantDR[i]$                      ▷ Step 2-5
10:     $response = QueryMatching(query, CurrDR[1...6])$                   ▷ Step 2-6
11:     $Send(response)$                                                  ▷ Step 2-7
```

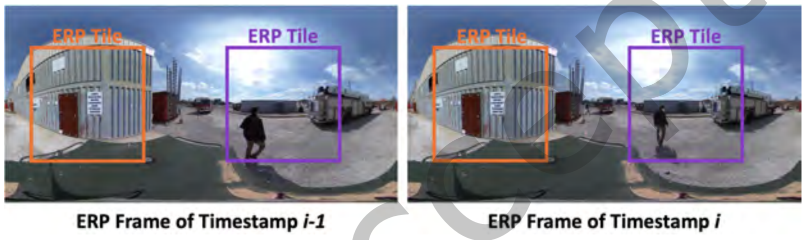

Fig. 8. An example of temporal redundancy. Network transmission and object detection processes are only necessary for the purple ERP Tile at timestamp *i*.

4.1 Motivation

Temporal filtering is motivated by the fact that significant inter-frame redundancy exists in video analytics tasks. For instance, surveillance and monitoring tasks generally deploy static cameras, meaning that the background landscape in the camera stream (e.g., ground and sky) may not constantly change with respect to time. Additionally, there are instances when the foreground content also remains unchanged for short periods, for example, during idle hours with no people or vehicles at a crossroad.

For example, the pair of images on Figure 8 indicates two ERP Frames in between neighboring timestamps $i-1$ and i, with two ERP Tiles labeled by purple and orange bounding boxes. In between the two timestamps, the object (human) in the purple ERP Tile has changed but not for the orange ERP Tile. Therefore, network transmission and object detection processes for the purple ERP Tile of timestamp i are necessary to discover the new position of the person, but would be unnecessary for the orange ERP Tile. It would be sufficient for the orange ERP Tile of timestamp i to take the detection result of the last timestamp.

Therefore, the transmission and detection processes are needed only when the objects within a ERP Tile change. However, detecting changes in objects between ERP Tiles is infeasible without performing object detection. Instead, we propose to use low-level image similarity metrics, such as Structural Similarity Index Measure (SSIM) and Peak Signal-to-Noise Ratio (PSNR), as potential indicators of object change between ERP Tiles. The PSNR or SSIM values of ERP Tiles can be computed at the edge. If a high similarity is identified, the transmission and detection processes for repetitive ERP Tiles can be bypassed to improve latency performance.

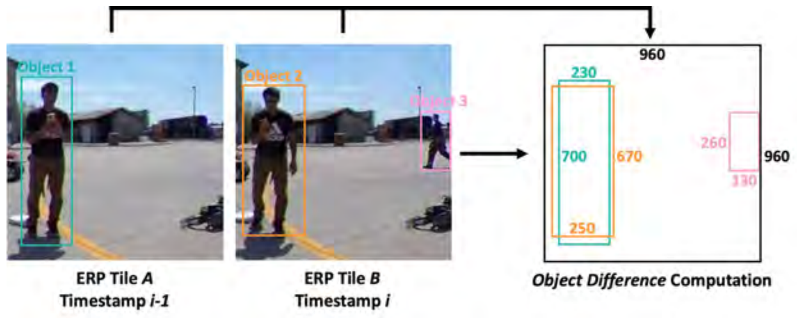

Fig. 9. An example showing the computation process of *object difference* between ERP Tile A of timestamp $i-1$ and ERP Tile B of timestamp i.

To evaluate the feasibility of this approach in terms of computation latency and accuracy, we need to answer the following two questions: 1) if low-level image similarity metrics between ERP Tiles can be computed efficiently on the edge device, and 2) if there is a negative correlation between low-level image similarity and object change in between ERP Tiles.

4.2 Object Difference

To explore the correlation between low-level image similarity and object change between ERP Tiles, a numerical metric is required to quantify the changes in objects between two ERP Tiles. Therefore, we have developed a new metric called *object difference*, which measures the relative difference of object bounding boxes between two ERP Tiles. Figure 9 and the following formulas summarize the computation process of *object difference*.

The computation of *object difference* takes in two arguments, O_A and O_B, which are the object detection results of ERP Tile A and ERP Tile B. Each of O_A and O_B is a set with each entry specifying the type and bounding box coordinates of an object appearing in the ERP Tile.

$$(MO_A, MO_B) = \left\{ (o_A, o_B) \in (O_A, O_B) \;\middle|\; \frac{\text{BBArea}(o_A) \cap \text{BBArea}(o_B)}{\text{BBArea}(o_A) \cup \text{BBArea}(o_B)} \geq 0.5, \text{Type}(o_A) = \text{Type}(o_B) \right\} \tag{1}$$

The computation process starts with Equation 1, where we find the object sets MO_A and MO_B based on O_A and O_B. Each object in MO_A matches exactly one object in MO_B. Two objects are determined to be matched if 1) they are of the same category, and 2) they have an Intersection over Union (IoU) equal or greater than 0.5. The IoU is calculated using the relative bounding box area, represented by BBArea(). Specifically, $\text{BBArea}(o_A)$ denotes the relative bounding box area of object o_A, calculated as the proportion of pixels within the bounding box of o_A to the total pixel count in the ERP Tile. For instance, Figure 9 presents the process of calculating *object difference* between ERP Tile A from timestamps $i-1$ and ERP Tile B from timestamp i. O_A contains Object 1 and O_B contains Object 2 and Object 3. Among them, Object 1 and Object 2 are matched because they are of the same type (human) and share a IoU of 0.884, which is higher than 0.5. Therefore, $(o_1, o_2) \in (MO_A, MO_B)$ while $o_3 \notin MO_B$.

Similarity Metric	RPi Latency (ms)	Laptop Latency (ms)
SSIM	$\mu = 339.4$ ($\sigma = 5.4$)	$\mu = 40.5$ ($\sigma = 0.5$)
PSNR	$\mu = 12.1$ ($\sigma = 0.7$)	$\mu = 6.1$ ($\sigma = 0.2$)

Table 1. Computation latency of SSIM and PSNR on the edge devices. Note that μ and σ represent the mean and standard deviation of the computation latency, respectively, calculated over 10 iterations of the experiment.

$$\text{objectDifference}(O_A, O_B) = \underbrace{\sum_{o_A \notin MO_A} \text{BBArea}(o_A) + \sum_{o_B \notin MO_B} \text{BBArea}(o_B)}_{part_1} \\ + \\ \underbrace{\sum_{(o_A, o_B) \in (MO_A, MO_B)} \Big[\Big(\text{BBArea}(o_A) \cup \text{BBArea}(o_B)\Big) - \Big((\text{BBArea}(o_A) \cap \text{BBArea}(o_B)\Big)\Big]}_{part_2} \tag{2}$$

In the next step illustrated by Equation 2, *object difference* of O_A and O_B is calculated by summing the relative bounding box areas of all non-matching objects ($part_1$). For the example in Figure 9, this refers to the relative bounding box area of Object 3, which is 0.0366. Then, the previous sum is added with the relative bounding box difference for all pairs of matching objects ($part_2$). The relative bounding box difference of two objects is determined by subtracting the intersection area from the union area. As in the case in Figure 9, the relative bounding box difference of Object 1 and Object 2 is 0.0220 and the final *object difference* is 0.0586.

4.3 Feasibility Study

To answer the two questions outlined in Section 4.1, we conducted the following feasibility study. We have used 1) a Raspberry Pi 4 (RPi) with a 4-core CPU of 1.8GHz and 2) a MacBook Pro laptop (Laptop) with a 4-core CPU of 2.6GHz for the edge device. To more accurately simulate real-world performance, a less powerful device (e.g., RPi) was chosen as the edge device, given that most commercial cameras lack powerful CPUs[28]. We have chosen a 10-minute ERP-formatted 360-degree video and segmented each ERP Frame into 6 ERP Tiles. For this study, we selected the human as the object of interest and manually labeled all instances in each ERP Tile. The *object difference* for each pair of neighboring ERP Tiles was calculated based on Equation 1 and Equation 2. The SSIM and PSNR values were computed for each pair of neighboring ERP Tiles on both edge devices. The experiment was conducted 10 times with the mean and standard deviation of the computation latency reported.

Table 1 lists the computation latency of SSIM and PSNR on the two edge devices. Note that computation latency is measured per ERP Frame (per 6 ERP Tiles). Figure 10 illustrates the distribution and Pearson correlation coefficient[3] between low-level image similarity values and *object difference* values after normalization. Each dot in the figure corresponds to a pair of these values, calculated from a pair of neighboring ERP Tiles. We notice that while both of the low-level similarity metrics can be computed efficiently on the Laptop, only the computation of PSNR finished within the satisfactory range on the RPi, with a latency of less than 100 milliseconds. Additionally, we found a Pearson correlation coefficient of less than -0.7 between PSNR and *object difference*, indicating a strong negative correlation. Given the performance observed on the RPi and the significant negative correlation, we believe that PSNR is the optimal estimator for object changes between ERP Tiles and should serve as the primary criterion for temporal filtering.

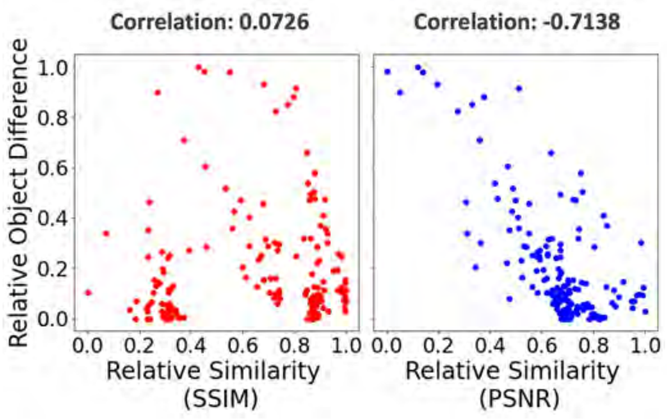

Fig. 10. Distribution and Pearson correlation coefficient[3] between low-level image similarity (SSIM and PSNR) values and *object difference* values after normalization.

Algorithm 3 Temporal Filtering Module

```
Input: $ET[1...6]$: The ERP Tile array received in the current timestamp
Initialize: $DominantET[1...6]$: The reference ERP Tiles for comparison against ERP Tiles in $ET$
            $WM[1...6]$: Weight matrices to correct pixels in ERP Tiles for geometric distortion
            $tt$: The similarity threshold
Output: $TFET[1...6]$: The temporal-compressed ERP Tile array of the current timestamp
        $TFSignal[1...6]$: The temporal filtering signal array for the ERP Tile array of
        the current timestamp
1:  for $i = 1, i++, i \leq 6$ do                          ▷ Iterating through the 6 ERP Tiles
2:      $ET[i] \leftarrow WM[i] \cdot ET[i]$                 ▷ Step 3-1
3:      $psnr_i \leftarrow ComputePSNR(ET[i], DominantET[i])$   ▷ Step 3-2
4:      if $psnr_i \geq tt$ then
5:          $TFET[i] \leftarrow Null$                       ▷ Step 3-3
6:          $TFSignal[i] \leftarrow False$                  ▷ Step 3-4
7:      else if $psnr_i < tt$ then
8:          $TFET[i] \leftarrow ET[i]$                      ▷ Step 3-5
9:          $TFSignal[i] \leftarrow True$                   ▷ Step 3-6
10:         $DominantET[i] \leftarrow ET[i]$                ▷ Step 3-7
11:     return $TFET[1...6], TFSignal[1...6]$             ▷ Step 3-8
```

4.4 Temporal Filtering Module

Following the insights gained from the feasibility study results, we have designed a module within the ST-360 framework to perform temporal filtering on ERP Tiles, as detailed in Algorithm 3. The temporal filtering module takes in ET, an array of 6 ERP Tiles received in the current timestamp, and returns the temporal-compressed ERP Tile array, $TFET$, as well as the corresponding temporal filtering signal array, $TFSignal$. The temporal filtering module operates with the $DominantET$ array, holding the previously captured ERP Tiles as reference points to which ERP Tiles in ET are compared. To reduce the numerical deviation due to geometric distortion, this module

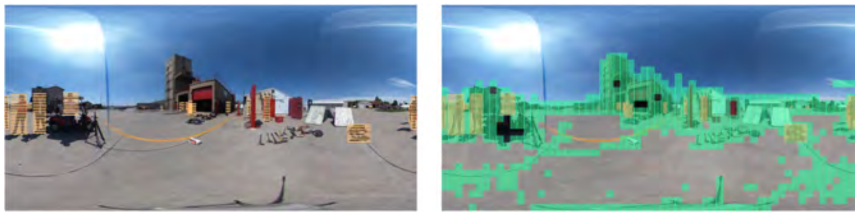

Fig. 11. The two images present the same example 360-degree ERP Frame with object-containing areas highlighted in orange. The spatial filtering results is annotated on the right image with the feature areas highlighted in green.

initializes an array of weight matrices, WM. These weight matrices are applied to each ERP Tile to assign less weight on highly distorted portions during PSNR computation. This correction process is fast because it requires only a single time of matrix multiplication. Furthermore, the weight matrices can be pre-computed to save online overhead.

Specifically, for a pixel in the ERP Tile, the correction it received is summarized in the following formula:

$$x' = x \cdot cos(\theta),$$

where x refers to a pixel in the ERP Tile and x' refers to x after correction. $cos(\theta)$ represents the corrected weight, which is negatively correlated to θ, the latitude of the pixel in the spherical representation. As previously discussed in Section 2.1, pixels on higher latitude suffer from more distortion, and hence are weighted less.

The temporal filtering module starts with Step 3-1 (line 2), where the currently received 6 ERP Tiles are multiplied by the weight matrices to correct for geometric distortion. Then, in Step 3-2, it calculates the PSNR value between each currently received ERP Tile ($ET[i]$) and its reference ERP Tile ($DominantET[i]$). When the PSNR value exceeds tt, the empirically set default similarity threshold of 37, this indicates a high degree of similarity between the two ERP Tiles. Under such circumstances, the corresponding ERP Tile at the current timestamp is entirely filtered out (Step 3-3), and a False boolean value is added to the corresponding entry in the temporal filtering signal array (Step 3-4), signaling the server component to bypass projection and detection for this ERP Tile. If the similarity threshold is not met, the ERP Tile remains uncompressed (Step 3-5), and a True boolean value is added to the corresponding entry in the temporal filtering signal array (Step 3-6). Meanwhile, in Step 3-7, the currently received ERP Tile replaces the reference ERP Tile in $DominantET$ and becomes the new reference point for later ERP Tiles. The results are returned at Step 3-8.

The complexity of Algorithm 3 is mainly determined by the computation of PSNR value between each pair of ERP Tiles (Step 3-2), which takes the complexity of $O(W \times H)$ for ERP Tiles of size $W \times H$. Therefore, the complexity of Algorithm 3 can be simplified as $O(W \times H)$.

5 SPATIAL FILTERING

5.1 Motivation

The spatial filtering module is motivated by the significant intra-frame redundancy found in 360-degree videos. For instance, the left image in Figure 11 shows an example of a 360-degree ERP-formatted frame, with areas containing objects highlighted in orange. In object-related queries, only these object-containing areas are of relevance. However, as illustrated in the example, areas without objects (e.g., sky and ground) occupy a considerable portion of the frame. Transmitting and detecting these non-object-containing areas results in suboptimal resource utilization. To enhance efficiency of the analytics system, it is essential to identify and compress these non-object-containing areas at the edge device before forwarding them to the server.

Feature	RPi Latency (ms)	Laptop Latency (ms)	Discovery Rate (%)	Reduction Rate (%)
Edge	$\mu = 74.4$ ($\sigma = 3.5$)	$\mu = 29.3$ ($\sigma = 2.7$)	93.5	40.5
Corner	$\mu = 252.1$ ($\sigma = 1.9$)	$\mu = 38.2$ ($\sigma = 2.3$)	91.8	45.6
Contour	$\mu = 3332.8$ ($\sigma = 3.8$)	$\mu = 1113.3$ ($\sigma = 17.1$)	42.9	81.0

Table 2. Computation latency, discovery rate, and reduction rate of different low-level image features achieved on the edge devices. Note that μ and σ represent the mean and standard deviation of the computation latency, respectively, calculated over 10 iterations of the experiment.

Low-level image features, such as edges, corners, or contours, have shown good performance in determining object-containing areas in 2D images[7, 28]. Furthermore, these features can be efficiently computed using existing feature extraction algorithms[28].

5.2 Feasibility Study

To evaluate the feasibility of this approach in terms of computation latency and accuracy, we conducted the following feasibility study to investigate 1) if low-level image feature extraction algorithms can be computed efficiently for ERP Tiles on the edge device, 2) if object-containing areas can be discovered by low-level image feature areas, and 3) if low-level image feature areas are substantially smaller than the overall tile areas.

The same hardware architecture and dataset described in Section 4.3 were used in this experiment. In addition, we selected 3 common low-level image features, including edge, corner, and contour, and computed feature pixels for each ERP Tile using the two edge devices (RPi and Laptop). The experiment was iterated 10 times with the mean and standard deviation of the computation latency reported. To identify feature areas, we empirically divided each ERP Tile into 32 x 32 grids. The feature areas are defined as the total areas of all grids that contain at least one feature pixel.

Table 2 presents the computation latency, discovery rate, and reduction rate of different features. Note that computation latency is measured per ERP Frame (per 6 ERP Tiles). Discovery rate is defined as the ratio of object areas included in feature areas. Ideally, a 100% discovery rate is desired to ensure that all potential object-containing areas are identified by feature extraction, thereby receiving higher image quality. Reduction rate is defined as the proportion of non-feature areas within the total tile area. A higher reduction rate is preferable, as it indicates that a greater ratio of non-feature areas can be downsampled or removed to save network transmission and object detection overhead.

The right image of Figure 11 presents a visual example when spatial filtering is applied to the 6 ERP Tiles of an 360-degree ERP Frame. The green areas represent the discovered feature areas and the orange areas represent the object-containing area. In this example, most of the object-containing areas are successfully discovered by the feature areas. Moreover, a considerable ratio of non-feature areas (e.g., sky and ground) are identified and can be downsampled.

Given the computational latency observed on the RPi and the discovery rate and reduction rate results, we believe that the edge feature is the optimal choice for spatial filtering.

5.3 Spatial Filtering Module

Based on the feasibility study results, we have designed the spatial filtering module for the ST-360 framework. This module operates by first extracting the edge feature pixels within each ERP Tile, followed by segmenting each ERP Tile into smaller grids. For each grid, it computes the feature density, which is defined to be the amount of edge feature pixels in each grid relative to the total pixel amount of the grid. Those grids with insufficient feature density are masked in black to optimize network transmission and object detection processes.

Algorithm 4 presents the detailed pipeline of the spatial filtering module. The spatial filtering module takes in ET, an array of 6 ERP Tiles captured at a timestamp, and returns the spatial-compressed ERP Tile array,

Algorithm 4 Spatial Filtering Module

```
Input:  $ET[1...6]$: The ERP Tile array received in the current timestamp
Initialize:  $ts$: The feature density threshold
             $(gw, gh)$: The grid size
Output:  $SFET[1...6]$: The spatial-compressed ERP Tile array of the current timestamp
 1: $gc \leftarrow ET.height/gh$                    ▷ Calculating the number of grids per column
 2: $gr \leftarrow ET.width/gw$                     ▷ Calculating the number of grids per row
 3: for $i = 1, i++, i \leq 6$ do                   ▷ Iterating through the 6 ERP Tiles
 4:     $GridET[1...gc][1...gr] \leftarrow SegmentToGrids(ET[i])$                    ▷ Step 4-1
 5:     for $m = 1, m++, m \leq gc$ do              ▷ Iterating through rows of grids
 6:         for $n = 1, n++, n \leq gr$ do          ▷ Iterating through columns of grids
 7:             $fpc \leftarrow CountPixel(CannyEdgeDetection(GridET[m][n]))$         ▷ Step 4-2
 8:             $gpc \leftarrow CountPixel(GridET[m][n])$
 9:             $fd \leftarrow fpc/gpc$                                               ▷ Step 4-3
10:             if $fd < ts$ then
11:                 $MaskWithBlack(GridET[m][n])$                                     ▷ Step 4-4
12:     $SFET[i] \leftarrow CombineGrids(GridET[1...gc][1...gr])$                    ▷ Step 4-5
13: return $SFET[1...6]$                                                              ▷ Step 4-6
```

$SFET$. This module initializes two variables, including 1) ts, the feature density threshold, which is empirically defaulted to 0.01, and 2) the grid size, which is empirically defaulted to 32 x 32. At Step 4-1 (line 4), each ERP Tile is segmented into grids of the specified size. These grids are then processed using the Canny Edge Detection algorithm[43], implemented via the OpenCV library, to identify feature pixels (Step 4-2). The feature density of each grid is calculated as the ratio of the number of feature pixels to the total number of pixels at Step 4-3. If the calculated feature density falls below the feature density threshold, this signifies a low presence of features in the grid. Consequently, the grid is marked entirely with black pixels at Step 4-4. Following this, at Step 4-5, each spatial-compressed ERP Tile is formed by recombining the grids. Finally, the spatial-compressed ERP Tile array is returned at Step 4-6.

The complexity of Algorithm 4 is primarily driven by the execution of the edge detection algorithm on each ERP Tile (Step 4-2), which requires $O(W \times H)$ time for an ERP Tile of size $W \times H$. Thus, the complexity of Algorithm 4 can be summarized as $O(W \times H)$.

6 END-TO-END EVALUATION

In this section, we present the comprehensive evaluation results of the ST-360 framework, focusing on end-to-end latency and query accuracy across various functional configurations.

6.1 Overview

6.1.1 ***Testing Environment.*** Three devices are selected to deploy the ST-360 framework, including: 1) a Raspberry Pi 4 (RPi) with a 4-core CPU of 1.8GHz, 2) a MacBook Pro laptop (Laptop) with a 4-core CPU of 2.6GHz, and 3) a desktop (Desktop) with a 6-core CPU and a NVIDIA GeForce RTX 3080 Ti graphics (12GB memory). The performance of the ST-360 framework is tested on the following combinations of hardware devices.

- Architecture 1: Edge: RPi; Server: Desktop
- Architecture 2: Edge: Laptop; Server: Desktop

As previously discussed in Section 4.3, RPi is chosen for the edge device because it has computational power comparable to that of normal commercial cameras. The edge device and the server device are wirelessly connected via Wi-Fi with an available bandwidth empirically configured to 20MBps to simulate the real-world network condition of remote surveillance and monitoring tasks.

6.1.2 ***Functional Configuration.*** We have chosen four schemas and three resolutions for comparative analysis, resulting in a total of 12 distinct functional configurations being tested. Each configuration represents a specific combination of a schema and a resolution.

The schemas include:

- FULL: Baseline schema with neither temporal nor spatial filtering module enabled;
- TF: Our filtering-based schema with only the temporal filtering module enabled;
- SF: Our filtering-based schema with only the spatial filtering module enabled;
- TFSF: Our filtering-based schema with both temporal and spatial filtering modules enabled.

The resolutions include:

- 4K: Each captured 360-degree ERP Frame is resized to 3840 x 1920;
- 2K: Each captured 360-degree ERP Frame is resized to 2880 x 1440;
- 1K: Each captured 360-degree ERP Frame is resized to 1920 x 960.

6.1.3 ***Latency Metric.*** End-to-end latency is calculated as the average delay per ERP Frame (per 6 ERP Tiles) across all 360-degree frames in the dataset. The delay per ERP Frame is defined as the time from when the edge camera captures the frame to when the server sends the query response to the VQS. We conduct the evaluation experiment 10 times and report the mean and standard deviation of the latency results.

6.1.4 ***Dataset and Object Detector.*** The evaluation is conducted based on a unique dataset of 14 one-minute 360-degree first-responders videos collected at a firefighting training institute using a RICOH THETA camera.

In our evaluation, we employ a YOLOv3 object detector[46] pre-trained on the COCO dataset[30] to identify objects. As discussed in Section 2.3, the dual-projection approach at 4K resolution achieves the best accuracy among all other configurations. Therefore, we first perform dual-projection and object detection across all 360-degree ERP Frames in the dataset at 4K resolution. The results serve as the ground-truth benchmark in our evaluation.

6.1.5 ***Query Task and Metric.*** We have chosen two query types for accuracy evaluation, including 1) object counting query and 2) object bounding box detection query. As discussed in Section 3.1, these two query types are supported by the Viewing and Query Service (VQS) of the ST-360 framework. Human is selected as the object of interest in our evaluation.

- Object Counting
  The object counting query tasks the analytics system with counting the amount of a specified category of object across all frames in the dataset. For a specific category of objects, c, the query accuracy is evaluated using *accuracy rate* defined by the following formula:

$$\textit{accuracy rate(c)} = \frac{1}{n}\sum_{i=1}^{n}\left(1 - \frac{|N^i_{pre-c} - N^i_{gt-c}|}{N^i_{gt-c}}\right),$$

  where n denotes the number of frames in the dataset, N^i_{pre-c} denotes the predicted amount of object c in frame i, and N^i_{gt-c} denotes the ground-truth amount of object c in frame i.
- Object Bounding Box Detection
  The object bounding box detection query tasks the analytics system with identifying the bounding box

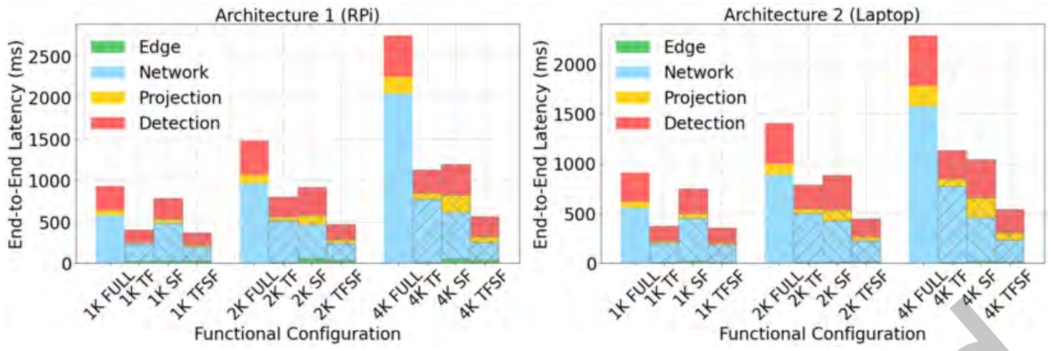

Fig. 12. Latency of each stage achieved by different functional configurations on Architecture 1 (RPi) and Architecture 2 (Laptop).

		1K			2K			4K		
		TF	SF	TFSF	TF	SF	TFSF	TF	SF	TFSF
Network	μ	63.3%	20.8%	70.4%	47.2%	57.5%	78.6%	63.1%	72.5%	89.7%
	σ	3.7%	0.6%	2.6%	3.3%	2.5%	2.1%	3.9%	3.0%	1.8%
Projection	μ	64.2%	10.4%	66.6%	60.5%	2.6%	61.9%	61.5%	0.8%	62.3%
	σ	0.7%	0.3%	1.9%	0.3%	<0.1%	1.1%	0.5%	<0.1%	0.7%
Detection	μ	48.0%	16.1%	49.0%	39.9%	16.1%	55.2%	43.1%	24.9%	52.4%
	σ	2.9%	0.1%	0.9%	0.9%	0.3%	0.5%	0.4%	0.2%	0.8%
Overall	μ	56.2%	15.6%	60.1%	45.7%	37.7%	68.3%	58.9%	56.5%	79.5%
	σ	3.0%	0.3%	1.7%	2.3%	1.1%	1.2%	2.6%	1.3%	1.4%

Table 3. Latency reduction on each stage achieved by different functional configurations for Architecture 1 (RPi). Note that μ and σ represent the mean and standard deviation of the latency reduction rate, respectively, calculated over 10 iterations of the experiment.

coordinates for every instance of a specified category of object across all frames in the dataset. The query accuracy of object bounding box detection is evaluated using standard metrics including Precision, Recall, and F1 score. The prediction of an object bounding box is determined to be correct when it achieves an Intersection over Union (IoU) score of 0.5 or higher compared to the ground-truth bounding box.

6.2 End-to-end Latency

Figure 12 displays the end-to-end latency results of the 12 functional configurations achieved on the two Architectures. Each plot details the individual latency contributors, including the spatial-temporal filtering stage (denoted as Edge), network transmission stage (denoted as Network), dual-projection stage (denoted as Projection), and object detection stage (denoted as Detection), for each functional configuration. Table 3 and Table 4 present the latency reductions in Network, Projection, and Detection stages achieved on functional configurations of filtering-based schemas (TF, SF, and TFSF) for Architecture 1 and Architecture 2 respectively.

We observed from Table 3 and Table 4 that functional configurations of filtering-based schemas lead to significant latency reductions in the Network, Projection, and Detection stages on both Architectures, resulting in a reduced end-to-end latency. Specifically, a consistent latency reduction performance is observed on TF-based functional configurations, achieving roughly a 50% latency reduction on the Network stage, 65% on the Projection stage, 45% on the Detection stage, and 50% for the end-to-end latency. In contrast, SF-based

		1K			2K			4K		
		TF	SF	TFSF	TF	SF	TFSF	TF	SF	TFSF
Network	μ	65.0%	22.1%	68.6%	44.2%	53.4%	75.9%	52.1%	71.4%	86.2%
	σ	0.8%	0.3%	3.7%	0.6%	0.5%	1.8%	0.5%	1.7%	0.8%
Projection	μ	66.2%	13.0%	68.6%	60.8%	3.4%	64.8%	64.7%	1.2%	64.8%
	σ	1.3%	0.2%	2.1%	2.1%	0.1%	2.0%	2.2%	<0.1%	2.6%
Detection	μ	46.4%	14.3%	48.4%	40.4%	15.2%	55.5%	44.3%	18.7%	52.4%
	σ	1.8%	0.4%	1.3%	1.3%	0.2%	2.1%	1.1%	0.7%	0.7%
Overall	μ	58.5%	17.0%	61.1%	43.9%	37.2%	68.2%	51.3%	52.5%	76.2%
	σ	1.4%	0.3%	2.5%	0.9%	0.5%	2.0%	0.8%	1.4%	1.3%

Table 4. Latency reduction on each stage achieved by different functional configurations for Architecture 2 (Laptop). Note that μ and σ represent the mean and standard deviation of the latency reduction rate, respectively, calculated over 10 iterations of the experiment.

	1K				2K				4K			
	FULL	TF	SF	TFSF	FULL	TF	SF	TFSF	FULL	TF	SF	TFSF
Precision	88.7%	88.5%	89.7%	89.2%	94.7%	95.6%	94.8%	95.4%	100.0%	96.0%	92.7%	94.2%
Recall	80.1%	80.3%	80.1%	79.1%	84.6%	82.2%	77.6%	77.7%	100.0%	96.0%	88.3%	87.0%
F1 Score	84.2%	84.2%	84.6%	83.9%	89.4%	88.4%	85.3%	85.6%	100.0%	96.0%	90.4%	90.5%
Acc. Rate	81.2%	81.3%	83.0%	81.5%	85.0%	78.2%	82.8%	78.7%	100.0%	87.2%	94.7%	86.4%

Table 5. Precision, Recall, and F1 score for object bounding box detection query, and accuracy rate for object counting query, achieved by different functional configurations.

functional configurations exhibit varying latency reduction performance across resolutions. In particular, the latency reduction in the Network stage improves from approximately 20% to 70%, while in the Detection stage, it ranges from about 15% to 25% as the resolution increases. Additionally, minimal latency reduction is observed in the Projection stage for SF-based functional configurations. While SF-based functional configurations achieve less latency reduction than TF-based functional configurations in most cases, we notice that they outperforms TF-based functional configurations in reducing the Network latency at 2K and 4K resolutions. This makes them a potentially better option for high-resolution 360-degree video transmission, especially in scenarios with limited bandwidth where network transmission is the primary contributor to the end-to-end latency. The TFSF-based functional configurations, which integrate the strengths of both spatial and temporal filtering modules, achieve the best latency performance in all individual stages across various resolutions.

For functional configurations of our filtering-based schemas, the Edge stage contributes minimally to the end-to-end latency on both Architectures. However, the filtering modules performed on the edge device significantly reduce latency in other time-intensive stages and in the overall end-to-end latency compared to functional configurations of the baseline FULL schema.

6.3 Accuracy

Table 5 provides a summary of the accuracy results for object bounding box detection query and object counting query, achieved by various functional configurations. Note that both Architectures yield identical accuracy results, as the choice of edge devices does not influence the results of object detection tasks executed on the server device.

In the object bounding box detection query, we observe a consistent trend across all four schemas: as resolution decreases, each accuracy metric—Precision, Recall, and F1 score—also diminishes. Notably, the Recall performance is about 10% lower than Precision for most functional configurations. At 1K resolution, functional configurations of all four schemas show very similar results in Precision, Recall, and F1 score. However, with the resolution

	1K				2K				4K			
	FULL	TF	SF	TFSF	FULL	TF	SF	TFSF	FULL	TF	SF	TFSF
Latency (μ) (ms)	927	406	782	370	1474	801	918	467	2746	1129	1193	563
Latency (σ) (ms)	28	22	17	10	19	41	26	9	119	51	28	10
F1 Score	84.2%	84.2%	84.6%	83.9%	89.4%	88.4%	85.3%	85.6%	100.0%	96.0%	90.4%	90.5%
Acc. Rate	81.2%	81.3%	83.0%	81.5%	85.0%	78.2%	82.8%	78.7%	100.0%	87.2%	94.7%	86.4%

Table 6. Summary of the evaluation with end-to-end latency (measured on Architecture 1), F1 score for the object bounding box detection query, and accuracy rate for the object counting query, achieved by different functional configurations. Note that μ and σ represent the mean and standard deviation of the latency, respectively, calculated over 10 iterations of the experiment.

increasing from 2K to 4K, the TF-based functional configurations consistently outperform the SF-based functional configurations in all accuracy metrics, particularly in Recall, where the TF-based functional configurations' performance approach those of baseline FULL-based functional configurations.

Similar to the object bounding box detection query, functional configurations of all four schemas demonstrate comparable accuracy at 1K resolution for the object counting query. Yet, with increasing resolution, the SF-based functional configurations outperform the TF-based functional configurations, contrasting with the trend observed in object bounding box detection query.

For both query types, the TFSF-based functional configurations exhibit accuracy performance comparable to the TF-based or SF-based configurations across all resolutions.

6.4 Evaluation Summary

Table 6 summarizes the evaluation results, detailing the end-to-end latency, F1 score for object bounding box detection query, and accuracy rate for object counting query, across different functional configurations and resolutions.

Relative to the FULL-based functional configurations, the TF-based functional configurations achieve roughly a 50% reduction in end-to-end latency while experiencing only a minor decrease in accuracy performance, with average less than a 5% drop in F1 score and a 7% decline in accuracy rate across resolutions. Conversely, the SF-based functional configurations fall short of the TF-based functional configurations in both end-to-end latency and object bounding box detection accuracy, yet they perform better in the object counting query. The TFSF-based functional configurations further enhance end-to-end latency without significantly compromising the accuracy, maintaining levels close to those of the TF-based and SF-based functional configurations. In particular, TFSF-based functional configurations achieve an end-to-end latency of less than 1000 milliseconds at all three resolutions, satisfying the latency requirement discussed in Section 2.2.

7 RELATED WORK

7.1 2D Video Analytics System

As outlined in Section 1, video analytics systems incorporate an edge camera that continuously captures and transmits video frames to a server for object detection. Given the constrained computational resources at the edge, existing analytics systems for 2D videos employ offloading strategies to relocate computation-intensive tasks like object detection to the server, which is equipped with more robust hardware capabilities. Recent years have primarily seen the proposal of two types of offloading strategies aimed at balancing the trade-off between latency and accuracy: compression-based strategies and partition-based strategies. Typical works of each type are discussed below.

In compression-based strategies, captured frames are compressed at the edge device to minimize network transmission latency, while the object detection process is offloaded to the server. For example, Jiang et al.[24] and

Guo et al.[17] proposed to adaptively compress frames at the edge device by adjusting configuration parameters, such as resolution and frame rate. Their approaches significantly lowered latency without compromising the detection accuracy. In Wang et al.'s work[59], the server identifies the background pixels within the current camera stream and forwards this information to the edge. The edge filters out these background pixels, transmitting only the foreground pixels back to the server for further analysis. Li et al.[28] introduced an offloading strategy in which the edge camera calculates frame differences using low-level image features and transmits only the distinct frames to the server for object detection. In the approach by Liu et al.[35], the edge camera discovers frame regions that potentially contain objects of interest, and masks the remaining regions prior to transmitting the frame to the server.

In partition-based strategies, the task of object detection is divided in between the edge and the server. For example, Wang et al.[62], Liu et al.[33], and Ran et al.[45] suggested deploying a compressed object detection model at the edge to enhance analytics speed. The studies by Wang et al.[62] and Liu et al.[33] involve the edge detecting larger objects through the compressed model, with regions containing smaller objects forwarded to the server for detailed detection. Ran et al.[45] implemented a decision engine at the edge to determine the optimal location for object detection, which considers factors including energy consumption, accuracy, and latency requirements.

The spatial and temporal filtering modules of our ST-360 framework was primarily inspired by the works of [28] and [35] because of their minimal computational demands on edge devices.

7.2 2D and 360-Degree Video Object Detection

Recent 2D object detectors can generally be classified into two types: 1) one-stage detectors, such as YOLO[46] and SSD[36], and 2) two-stage detectors, including R-CNN[15], Fast RCNN[14], Faster R-CNN[47], and FPN[31]. Because of the end-to-end latency requirements, we have utilized the faster one-stage detectors (YOLO) to construct our 360-degree video analytics framework.

Meanwhile, several studies have introduced object detection frameworks customized for 360-degree videos. For instance, based on Faster R-CNN, Su et al.[54] developed the first spherical convolutional neural network (S-CNN) designed for object detection, which is capable of feature extraction from spherical 360-degree images. Yu et al.[73] and Wang et al.[60] each proposed their unique 360-degree object detectors, utilizing R-CNN and S-CNN, respectively, to directly process 360-degree images in ERP format. However, the majority of these 360-degree object detection frameworks are derived from two-stage detectors, which require extended processing time and, therefore, do not align with our low-latency requirements. Moreover, detectors like those mentioned in [60, 67, 73] continue to encounter geometric distortion.

7.3 360-Degree Video Projection

Table 7 details the primary projection methods utilized by 360Lib[19], a library from the High Efficiency Video Coding (HEVC)[55] and Versatile Video Coding (VVC)[5] encoders. These projection methods are widely used in 360-degree video compression and streaming, which is a key component in 360-degree video analytics systems.

Among all projection methods, equirectangular projection (ERP) is extensively employed for 360-degree video storage and transmission because ERP-formatted videos fit in 2D video codec well. However, there is a trade-off involved as equirectangular projection introduces geometric distortion, which can have a negative impact on downstream visual tasks such as object detection. Contrary to the equirectangular projection that maps spherical content onto a single flat plane, cubemap projection (CMP) method projects the 3D spherical environment onto the six faces of a cube. This approach significantly reduces geometric distortion, especially at the center of each cube face, which enhances the accuracy of downstream visual tasks.

Cylindrical Projections	**Azimuthal Projections**	**Other Projections**
Cylindrical Projection [38]	Orthographic Projection [23]	Cubemap Projection [11]
Mercator Projection [16]	Stereographic Projection [32]	Sinusoidal Projection [50]
Miller Projection [53]	Fisheye Projection [49]	Transverse Mercator Projection [40]
Equirectangular Projection [39]	Equisolid Projection [41]	Pannini Projection [51]
Lambert Cylindrical Equal Area [57]		Architectural Projection [4]
		Viewport Projection [76]

Table 7. Existing 360-degree video projection methods.

8 CONCLUSION

In this work, we introduce ST-360, the first online video analytics framework tailored for 360-degree videos. The ST-360 framework utilizes the dual-projection approach to reduce geometric distortion, and incorporates the novel spatial and temporal filtering modules to effectively optimize the network transmission, dual-projection, and object detection processes. Our evaluation, utilizing a unique dataset of 14 360-degree videos, demonstrates that our filtering-based functional configurations (TF, SF, and TFSF) achieve an average reduction in end-to-end latency of over 50% compared to the baseline configuration, while maintaining high accuracy in both object counting and object bounding box detection queries.

ACKNOWLEDGMENTS

We highly appreciate the support from firefighters and researchers at the Illinois Fire Service Institute (IFSI), especially Dr. Farzaneh Masoud and Dr. Dac Nguyen, for helping us collect the dataset. This work was funded by National Science Foundation grants NSF 2140645, NSF 2140620, NSF 1835834, NSF 2106592, and NSF 1900875.

REFERENCES

[1] Md Adnan Arefeen, Sumaiya Tabassum Nimi, and Md Yusuf Sarwar Uddin. 2022. FrameHopper: Selective Processing of Video Frames in Detection-driven Real-Time Video Analytics. arXiv:2203.11493 [cs.CV]
[2] Roberto G de A Azevedo et al. 2019. Visual distortions in 360° videos. *TCSVT* 30, 8 (2019), 2524–2537.
[3] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. *Pearson Correlation Coefficient.* Springer Berlin Heidelberg.
[4] Andrew E Benjamin. 2012. *Architectural projections.* RMIT University Press Australia.
[5] Benjamin Bross et al. 2021. Overview of the Versatile Video Coding Standard and its Applications. *TCSVT* (2021).
[6] Bo Chen et al. 2019. Event-Driven Stitching for Tile-Based Live 360 Video Streaming. In *MMSys*.
[7] Bo Chen et al. 2022. Context-aware image compression optimization for visual analytics offloading. In *MMSys*.
[8] Jueyu Chen et al. 2023. Boundary-aware Shadow Detection via Mask Decoupling and Feature Correction. In *ICME*.
[9] Benjamin Coors et al. 2018. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *ECCV*. 518–533.
[10] Mallesham Dasari et al. 2020. Streaming 360-Degree Videos Using Super-Resolution. In *INFOCOM*.
[11] Fanyi Duanmu et al. 2018. Hybrid cubemap projection format for 360-degree video coding. In *DCC*.
[12] Ching-Ling Fan et al. 2017. Fixation Prediction for 360° Video Streaming in Head-Mounted Virtual Reality. In *NOSSDAV*.
[13] Ching-Ling Fan et al. 2019. A survey on 360 video streaming: Acquisition, transmission, and display. *Csur* 52, 4 (2019).
[14] Ross Girshick. 2015. Fast R-CNN. In *ICCV*. 1440–1448.
[15] Ross Girshick et al. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.
[16] E Grafarend. 1995. The optimal universal transverse Mercator projection. In *Geodetic Theory Today: Third Hotine-Marussi Symposium on Mathematical Geodesy L'Aquila, Italy, May 30–June 3, 1994*. Springer, 51–51.
[17] Junpeng Guo et al. 2022. VPPlus: Exploring the Potentials of Video Processing for Live Video Analytics at the Edge. In *IWQoS*. 1–11.
[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence* 37, 9 (2015), 1904–1916.
[19] Yuwen He and the JVET team. 2021. 360Lib: Add-on Library for Handling 360 Degree Panoramic Projection Formats. https://vcgit.hhi.fraunhofer.de/jvet/360lib Accessed: 2024-07-04.

[20] Mohammad Hosseini and Viswanathan Swaminathan. 2016. Adaptive 360 VR video streaming: Divide and conquer. In *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE, 107–110.

[21] Zhirui Hu et al. 2022. Quantum neural network compression. In *DAC*.

[22] Jingwei Huang, Zhili Chen, Duygu Ceylan, and Hailin Jin. 2017. 6-DOF VR videos with a single 360-camera. In *2017 IEEE Virtual Reality (VR)*. IEEE, 37–44.

[23] Thomas S. Huang and Chia-Hoang Lee. 1989. Motion and structure from orthographic projections. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11, 5 (1989), 536–540.

[24] Junchen Jiang et al. 2018. Chameleon: Scalable Adaptation of Video Analytics. In *SIGCOMM*.

[25] Mohammad Khalil, Paul Prinsloo, and Sharon Slade. 2022. A comparison of learning analytics frameworks: A systematic review. In *LAK22: 12th international learning analytics and knowledge conference*. 152–163.

[26] Chaoyang Li et al. 2023. Concerto: Client-server Orchestration for Real-Time Video Analytics. In *Proceedings of the 31st ACM International Conference on Multimedia (MM)*.

[27] Jiaxi Li et al. 2023. Latency-Aware 360-Degree Video Analytics Framework for First Responders Situational Awareness. In *NOSSDAV*.

[28] Yuanqi Li et al. 2020. Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics. In *SIGCOMM*.

[29] Jingwei Liao et al. 2021. Shadow detection via predicting the confidence maps of shadow detection methods. In *MM*.

[30] Tsung-Yi Lin et al. 2015. Microsoft COCO: Common Objects in Context. arXiv:1405.0312 [cs.CV]

[31] Tsung-Yi Lin et al. 2017. Feature pyramid networks for object detection. In *CVPR*.

[32] Richard J Lisle and Peter R Leyshon. 2004. *Stereographic projection techniques for geologists and civil engineers*. Cambridge University Press.

[33] Luyang Liu, Hongyu Li, and Marco Gruteser. 2019. Edge Assisted Real-Time Object Detection for Mobile Augmented Reality. In *The 25th Annual International Conference on Mobile Computing and Networking*. Article 25, 16 pages.

[34] Peng Liu, Bozhao Qi, and Suman Banerjee. 2018. Edgeeye: An edge service framework for real-time intelligent video analytics. In *Proceedings of the 1st international workshop on edge systems, analytics and networking*. 1–6.

[35] Shengzhong Liu et al. 2022. AdaMask: Enabling Machine-Centric Video Streaming with Adaptive Frame Masking for DNN Inference Offloading. In *Proceedings of the 30th ACM International Conference on Multimedia*. 3035–3044.

[36] Wei Liu et al. 2016. SSD: Single Shot MultiBox Detector. In *ECCV*.

[37] Xianwei Lv, Qianqian Wang, Chen Yu, and Hai Jin. 2023. A Feedback-Driven DNN Inference Acceleration System for Edge-Assisted Video Analytics. *IEEE Trans. Comput.* 72, 10 (2023), 2902–2912.

[38] Osborn Maitland Miller. 1942. Notes on cylindrical world map projections. *Geographical Review* 32, 3 (1942), 424–430.

[39] Ronald Miller. 1949. An Equi-Rectangular Map Projection. *Geography* (1949), 196–201.

[40] Laurence Moore. 1997. Transverse mercator projections and us geological survey digital products. *US Geological Survey, Professional Paper* (1997).

[41] Julien Moreau, Sébastien Ambellouis, and Yassine Ruichek. 2013. Equisolid fisheye stereovision calibration and point cloud computation. In *ISPRS-SSG 2013, conference on Serving Society with Geoinformatics*. 6p.

[42] Anh Nguyen and Zhisheng Yan. 2023. Enhancing 360 Video Streaming through Salient Content in Head-Mounted Displays. *Sensors* 23, 8 (2023).

[43] OpenCV. 2023. Canny Edge Detection. https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html Accessed: 2024-06-14.

[44] Jounsup Park. 2021. Real-time object detection in 360-degree videos. In *Real-Time Image Processing and Deep Learning 2021*. 99–114.

[45] Xukan Ran et al. 2018. DeepDecision: A Mobile Deep Learning Framework for Edge Video Analytics. In *INFOCOM*.

[46] Joseph Redmon et al. 2016. You only look once: Unified, real-time object detection. In *CVPR*.

[47] Shaoqing Ren et al. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Neurips* 28 (2015).

[48] Ayush Sarkar et al. 2022. A 360-Degree Video Analytics Service for In-Classroom Firefighter Training. In *CPS-ER*.

[49] Ellen Schwalbe. 2005. Geometric modelling and calibration of fisheye lens camera systems. In *Proc. 2nd Panoramic Photogrammetry Workshop, Int. Archives of Photogrammetry and Remote Sensing*, Vol. 36. Citeseer, W8.

[50] Jeong Chang Seong, Karen A Mulcahy, and E Lynn Usery. 2002. The sinusoidal projection: A new importance in relation to global image data. *The Professional Geographer* 54, 2 (2002), 218–225.

[51] Thomas K Sharpless, Bruno Postle, and Daniel M German. 2010. Pannini: A New Projection for RenderingWide Angle Perspective Images.. In *CAe*. 9–16.

[52] Jangwoo Son and Eun-Seok Ryu. 2018. Tile-based 360-degree video streaming for mobile virtual reality in cyber physical system. *Computers & Electrical Engineering* 72 (2018), 361–368.

[53] William H Sprinsky and John P Snyder. 1986. The Miller oblated stereographic projection for Africa, Europe, Asia and Australasia. *The American Cartographer* 13, 3 (1986), 253–261.

[54] Yu-Chuan Su et al. 2017. Learning spherical convolution for fast features from 360 imagery. *Neurips* 30 (2017).

[55] Gary J. Sullivan et al. 2012. Overview of the High Efficiency Video Coding (HEVC) Standard. *TCSVT* (2012).

[56] Atsushi Tagami et al. 2019. Tile-Based Panoramic Live Video Streaming on ICN. In *ICC Workshops*.

[57] WR Tobler. 1973. The hyperelliptical and other new pseudo cylindrical equal area map projections. *Journal of Geophysical Research* 78, 11 (1973), 1753–1759.
[58] Fu-En Wang, Yu-Hsuan Yeh, Min Sun, Wei-Chen Chiu, and Yi-Hsuan Tsai. 2020. Bifuse: Monocular 360 depth estimation via bi-projection fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 462–471.
[59] Hanling Wang et al. 2023. VaBUS: Edge-Cloud Real-Time Video Analytics via Background Understanding and Subtraction. *IEEE Journal on Selected Areas in Communications* (2023).
[60] Kuan-Hsun Wang et al. 2019. Object detection in curved space for 360-degree camera. In *ICASSP*.
[61] Wufan Wang, Bo Wang, Lei Zhang, and Hua Huang. 2022. Sensitivity-Aware Spatial Quality Adaptation for Live Video Analytics. *IEEE Journal on Selected Areas in Communications* 40, 8 (2022), 2474–2484.
[62] Xu Wang et al. 2021. EdgeDuet: Tiling Small Object Detection for Edge Assisted Autonomous Mobile Vision. In *INFOCOM*.
[63] Ziyi Wang et al. 2023. Edge-Assisted Real-Time Video Analytics With Spatial–Temporal Redundancy Suppression. *IoT-J* (2023).
[64] Lan Xie et al. 2017. 360ProbDASH: Improving QoE of 360 Video Streaming Using Tile-Based HTTP Adaptive Streaming. In *MM*.
[65] Mai Xu et al. 2020. State-of-the-art in 360 video/image processing: Perception, assessment and compression. *IEEE J-STSP* (2020).
[66] Zhisheng Yan and Jun Yi. 2022. Dissecting Latency in 360° Video Camera Sensing Systems. *Sensors* 22, 16 (2022), 6001.
[67] Junhuan Yang et al. 2022. Automated Architecture Search for Brain-inspired Hyperdimensional Computing. *arXiv preprint arXiv:2202.05827* (2022).
[68] Junhuan Yang et al. 2023. On-Device Unsupervised Image Segmentation. *arXiv preprint arXiv:2303.12753* (2023).
[69] Peng Yang et al. 2019. Edge coordinated query configuration for low-latency and accurate video analytics. *TII* (2019).
[70] Wenyan Yang et al. 2018. Object detection in equirectangular panorama. In *icpr*. IEEE, 2190–2195.
[71] Abid Yaqoob et al. 2020. A survey on adaptive 360 video streaming: Solutions, challenges and opportunities. *IEEE Communications Surveys & Tutorials* 22, 4 (2020), 2801–2838.
[72] Shanhe Yi et al. 2017. Lavea: Latency-aware video analytics on edge computing platform. In *SEC*.
[73] Dawen Yu and Shunping Ji. 2019. Grid based spherical cnn for object detection from panoramic images. *Sensors* 19, 11 (2019), 2622.
[74] Tingting Yuan, Liang Mi, Weijun Wang, Haipeng Dai, and Xiaoming Fu. 2023. AccDecoder: Accelerated Decoding for Neural-enhanced Video Analytics. arXiv:2301.08664 [cs.CV]
[75] Alireza Zare et al. 2016. HEVC-compliant viewport-adaptive streaming of stereoscopic panoramic video. In *PCS*. 1–5.
[76] Alireza Zare et al. 2017. Virtual reality content streaming: Viewport-dependent projection and tile-based techniques. In *ICIP*.
[77] Miao Zhang et al. 2022. CASVA: Configuration-Adaptive Streaming for Live Video Analytics. In *INFOCOM*.
[78] Miao Zhang et al. 2023. OmniSense: Towards Edge-Assisted Online Analytics for 360-Degree Videos. In *INFOCOM*.
[79] Rui-Xiao Zhang et al. 2023. Owl: A Pre-and Post-processing Framework for Video Analytics in Low-light Surroundings. In *INFOCOM*.