A Contributory Public-Event Recording and Querying System

Arun Joseph
Indian Institute of Science
Bangalore, India
arunj@iisc.ac.in

Nikita Yadav
Indian Institute of Science
Bangalore, India
nikitayadav@iisc.ac.in

Vinod Ganapathy
Indian Institute of Science
Bangalore, India
vg@iisc.ac.in

Dushyant Behl
IBM Research
Bangalore, India
dushyantbehl@in.ibm.com

ABSTRACT

CCTV (Closed-Circuit Television) systems are commonly used for security and surveillance. They provide a visual record of events, which can be used to monitor criminal activity, support investigations, and improve public safety. Many cities have implemented a number of cameras for surveillance, with Delhi, India having 1446 cameras per square mile. These cameras are installed mainly by official traffic police or city authorities, but private organizations and individuals also have their cameras pointed toward public spaces. In many cases, the city authorities or police require query capability and access of the video feed from these CCTV cameras to perform diligent inquiries but the private entities usually do not share the raw footage due to various concerns.

To address the above issue, we introduce the "Public Event Recording and Querying System (PERQS)", a video analysis-based querying system. PERQS is a contributory CCTV network that encompasses both private and public parties. PERQS provides a reliable and secure solution for querying CCTV video feeds by leveraging video analytic algorithms. It ensures video privacy by allowing participants to perform video analysis locally on their servers. Additionally, PERQS employs hash-based commitments and query consensus, guaranteeing tamper-proof and accurate results. Furthermore, it provides a novel consensus mechanism called time-travel consensus to build trust in the query output even in the case of a lack of cameras pointing at a particular location. We introduce a query language PERQL, similar to SQL, which makes the system expandable and flexible. The plug-and-play architecture allows for integration with advanced vision models and algorithms for analysis.

CCS CONCEPTS

• **Information systems** → *Collaborative and social computing systems and tools.*

SEC '23, December 6–9, 2023, Wilmington, DE, USA

ACM ISBN 979-8-4007-0123-8/23/12…$15.00
https://doi.org/10.1145/3583740.3628445

KEYWORDS

contributory-CCTV, video analytics, video querying, blockchain

ACM Reference Format:
Arun Joseph, Nikita Yadav, Vinod Ganapathy, and Dushyant Behl. 2023. A Contributory Public-Event Recording and Querying System. In *The Eighth ACM/IEEE Symposium on Edge Computing (SEC '23), December 6–9, 2023, Wilmington, DE, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3583740.3628445

1 INTRODUCTION

CCTV (Closed-Circuit Television) technology has become an invaluable asset for security and surveillance in various settings, including homes, businesses, public areas, and critical infrastructure. These systems consist of cameras, recorders, and monitors that enable real-time monitoring and recording of events. CCTV's significance stems from its ability to provide a visual record of events. It serves purposes such as deterring and monitoring criminal activity, furnishing evidence for criminal investigations, and enhancing public safety [1, 4, 24, 36, 38, 46, 49, 56]. CCTV use has increased in recent years due to technological advancements and increased demand for security and surveillance [3, 5].

The statistics in Table 1 reveal that Delhi, India, has 1446 cameras per square mile. These cameras are primarily installed by official traffic police and city authorities. In addition, many private organizations and individuals also have their own CCTV cameras which look out at public spaces. In London, there are over 942,562 CCTV cameras if we include private cameras [11], meaning there is one CCTV camera for every ten people in the city. A person is likely to be captured on London CCTV up to 70 times a day [1, 11]. Despite the abundance of CCTV cameras in neighborhoods, utilizing them collectively poses several complexities. The video streams captured by these cameras are owned by various entities, both private and public, making it difficult to query and access them as a unified resource.

In some situations, public authorities require access to additional footage captured by private cameras to aid their investigations. Let us consider a scenario where a car is involved in an accident and flees the scene. The identification of the vehicle becomes crucial in apprehending the culprit. Law enforcement agencies examine the traffic police CCTV footage, but if the captured view is limited, they explore additional sources, such as cameras installed by nearby businesses or residents. By analyzing multiple footage sources, authorities can cross-reference information to verify the registration number of the fleeing vehicle. This highlights the importance of

Table 1: Number of CCTV cameras in some of the world's populated cities [2].

City	# Cameras	Per Square Miles	Per 1000 People
Delhi, India	436600	1446	26.7
Chennai, India	282126	614	24.53
Singapore, Singapore	108981	281	18.04
Seoul, South Korea	77814	333	7.8
Moscow, Russia	213000	219	16.85
London, UK	127373	210	13.35
New York, USA	56190	187	6.87
Cities of China[1]	54000000	1000+	372+

a ***contributory CCTV networks*** in assisting law enforcement in solving crimes and maintaining public safety.

Participating in contributory CCTV systems provides several benefits to organizations, both public and private. The main objective of CCTV is to enhance security and ensure the neighborhood's safety. By sharing public CCTV footage, the overall surveillance and security of the area can be significantly improved. In cases where one camera may be damaged, malfunctioning, or experiencing technical issues, other cameras from different organizations can still monitor the area, ensuring continuous surveillance coverage.

Building such a comprehensive contributory CCTV network system poses several challenges. Private entities are often reluctant to share their video feeds with public authorities. Concerns regarding privacy, data security, and trust in the appropriate use of footage contribute to this hesitation. Additionally, the video feed's reliability, camera quality variations, time synchronization, and blind spots in surveillance coverage make it difficult to create an effective system. The four significant challenges in designing such a system are:

① **Untrustworthy Video Feed:** Video feeds are compromised by technical errors or malicious activities, resulting in damaged or edited footage that is unreliable for investigations. Moreover, a lack of trust exists among organizations, making video footage sharing difficult. There is always the risk that organizations might intentionally share incorrect footage to benefit themselves or to deceive others.

② **Privacy of Video Feed:** Private organizations do not wish to share raw video feeds. They are concerned about the security and privacy of data, as well as the potential for unauthorized access and misuse of the footage. These concerns make it difficult to get all parties to participate in a contributory CCTV network.

③ **Variable Quality of Video Feed:** The cameras used for recordings differ in quality, viewing angle, aspect ratio, and color calibration, making it difficult to standardize and integrate the footage for analysis. Additionally, the recording timestamps of different cameras may not be perfectly synchronized, leading to inconsistencies in the data. Moreover, the sheer volume of data generated by multiple cameras will make it challenging to process and analyze the footage efficiently.

④ **Availability of CCTV Footage:** The absence of CCTV cameras in certain areas results in blind spots, leaving surveillance systems incomplete. Additionally, camera malfunctions also lead to gaps in footage. Even without direct visual evidence, it is possible to use nearby cameras to infer what happened. However, identifying the appropriate neighboring cameras and effectively utilizing them is a challenging task.

Prior works related to video analytics and querying are not equipped to solve many of these challenges. [7, 8, 28, 29, 40, 58, 60] can be used to query large set of videos with the help of video analytic tools. They assume videos are readily available for analysis. None of them addresses video privacy, correctness, or mutual trust between participants. While [13, 39, 51, 52, 59] attempt to address the privacy issues to an extent, none of the prior work attempts to build a collaborative CCTV network that respects participants' privacy and can handle mutual distrust between participants. A detailed related works discussion can be found in Section 7.

In this work, we introduce an innovative system called the ***Public Event Recording and Querying System (PERQS)***. Our system aims to establish an efficient distributed network of CCTV cameras, overcoming the need for participants to place blind trust in one another. The core idea behind PERQS is to create a collaborative environment where users submit queries to the system, and it will autonomously utilize the entire network of camera feeds to provide responses by leveraging advanced video analytic techniques. This eliminates the need for users to review individual camera feeds manually or rely on the cooperation of specific participants within the network. PERQS introduces a novel consensus based video analysis concept to solve trust and privacy challenges associated with a contributory CCTV camera network. The insights we put together to solve the main challenges are as follows.

① **Tamper-proof Video feed:** PERQS introduces video-hash commitment and verification, which assures video authenticity. In case of disputes, the authenticity of the video footage can be easily verified, eliminating any concerns about the video being edited or tampered with later on. This added level of security and transparency further supports the importance of participating.

② **Private Video Storage:** PERQS does not share the raw video feeds. Instead, the system only shares the video analysis results, ensuring that sensitive information is kept private. This approach enhances privacy and eliminates the need for a vast central storage facility to store massive amounts of video data.

③ **Consensus-based Video Analysis:** To address the reliability issues of individual cameras, PERQS uses consensus, where multiple cameras participate in a query execution to reach a result. This approach ensures that the camera's fault, damage, or malicious behavior does not affect the output. Consensus not only improves the accuracy of the surveillance but also reduces the likelihood of false positives.

④ **Time-travel Consensus:** If there are blind spots in the city or camera faults, we devised a novel time-travel consensus mechanism in PERQS that uses footage from other cameras to answer the query. This mechanism can also be used when there are insufficient

[1]Due to limited transparency, the exact number of cameras in each Chinese city is still being determined. However, if China has 540 million cameras and divides it among its 1.46 billion population, we can estimate that there are approximately 373 cameras per 1,000 people.

Figure 1: An example city map with CCTV cameras. Red cameras are the traffic police cameras, blue are the institution cameras, and green are homeowner's cameras. The map pointer represents the GPS location of these cameras. The intersections are marked on the road with alphabets.

cameras to reach a consensus. The system uses nearby cameras to access pre-event or post-event footage to achieve consensus. This process is known as time-travel consensus because it enables the system to "travel through time" and use footage from other cameras to fill in gaps in the recordings.

The requirements of mutual distrust between participants, the hash-based commitment of video feeds, and the consensus naturally led us to adopt a blockchain-based architecture. This also helps us to ensure that participants do not need to trust each other for either the video feeds or the analysis results. Consensus and hash verification enable us to establish collective correctness, ensuring the accuracy and reliability of the system. To enable a flexible and customizable user experience, we developed the PERQL query language, similar to SQL, an extension of FRAMEQ[28]. This language allows users to design queries tailored to their specific needs while also providing the ability to combine multiple queries to answer more complex questions. Each participant maintains their analysis facility for the video analysis, and our architecture ensures that any video analysis tool can be easily plugged into the system. We combine the novel query consensus, time-travel consensus, tamperproof videos, and query language in PERQS to address all of the challenges previously discussed.

The novel contributions of this paper are:

- Design of a distributed collaborative CCTV video querying framework, **PERQS**, its prototype implementation and evaluation.
- ***Query consensus*** mechanism to solve mutual trust and video quality-related issues.
- ***Time-travel consensus*** to improve the query consensus with the help of additional footage from nearby cameras.

While the PERQS system has several advantages, it also has inherent limitations. One of the significant limitations is its inability to perform joint video comparisons effectively. Various vision techniques are available to compare and analyze the two videos jointly [33, 36, 55, 57, 60]. However, since PERQS does not share videos, applications requiring joint video analysis are not feasible.

The rest of the paper is as follows: we explain the use cases which motivated the design of PERQS in Sec 2, we present the design of our PERQS system in Sec 3, we explain our query execution engine in more detail along with our query language PERQL in Sec 4, we explain our implementation in Sec 5, we evaluate our system in Sec 6, discuss related work in Sec 7 and finally discuss future work and conclude the paper in Sec 8

2 EXAMPLE USE CASES

A contributory network of CCTV(s) can help a lot in surveillance and security. We present some practical examples where pooling will help us to get better and more accurate results. We use the map in Figure 1 to illustrate the examples.

Example 1: Hit and Run Incident. A car involved in an accident flees the scene. The objective is to locate the culprit by identifying the car involved. Consider the map in Figure 1. Suppose the accident happened at junction *'D'*, and the traffic police CCTV nearby only captured a side view of the vehicle. According to the figure, the car originated from somewhere between *'B'* and *'D'* and proceeded towards *'E'*. Unfortunately, there are no other traffic police cameras available to identify the license plate number. However, there is a camera at junction *'C'* and a few individual cameras on the way towards *'A'*. By utilizing this supplementary footage, we can pinpoint the individuals responsible for the accident.

Example 2: Incident in a Camera Blind Area. Consider a scenario where two vehicles collide in an area without CCTV cameras. Suppose that one vehicle's excessive speed causes an accident. Video footage can be gathered from homeowners, shop owners, and ATMs along the road to estimate the car's speed. Combined, these pieces of evidence can support the claim of an overspeeding accident. Say a truck travelled from *'B'* to *'M'*, a car from *'M'* to *'C'* and the accident occurred between *'D'* and *'I'*. There are a couple of traffic police cameras along the road and many other private cameras. They can all participate in the over-speed estimation process.

Example 3: Vehicle counting. Counting the number of cars on a particular road is a critical task for traffic management and infrastructure planning. This information helps authorities assess traffic conditions, identify congested areas, and make informed decisions about road widening or constructing additional routes to alleviate traffic congestion. However, challenges arise when the cameras are positioned on the side of the road, as they may not capture all passing vehicles. For example, if a camera is focused on one side of the road, it may not detect cars traveling alongside large vehicles, such as buses or trucks. This limitation can lead to undercounting and inaccurate representation of the traffic volume. Leveraging multiple cameras in this scenario can provide a higher level of accuracy and confidence in the car count.

Example 4: Road Safety Violation. Traffic safety is a vital concern in every city and town around the world. In order to keep roads safe for drivers, passengers, pedestrians, and cyclists, it is essential to enforce traffic rules and regulations. Accidents can occur when vehicles fail to adhere to traffic safety regulations, such as running red lights, speeding, or riding a bike without wearing a helmet. It is crucial to identify these violations and issue fines or warnings to those responsible. While traffic police cameras may not provide comprehensive coverage of all areas within the city, by collaborating with other stakeholders in the community, traffic police can access additional footage from these privately owned cameras to help identify and apprehend traffic violators more accurately. This additional information can be used to investigate accidents, determine who is at fault, and enforce traffic laws. Ultimately, a collaborative approach to surveillance can lead to safer streets, fewer accidents, and a better quality of life for everyone in the community.

As we have observed, the pooling of CCTV footage from a variety of sources has the potential to enhance public safety greatly. However, there are also several challenges that come with bringing together footage from private, public, and individual organizations. We present PERQS (Public Event Recording and Querying System) as a solution to the challenges of pooling CCTV footage from various organizations. The detailed system design is explained in the next sections.

3 PERQS DESIGN

PERQS allows public, private, and individual CCTV cameras to be connected and integrated into a contributory distributed network. Organizations that own one or more public CCTV cameras can participate in this contributory network. As discussed earlier, such a system poses many challenges in ensuring the trustworthiness of video feeds, participants' privacy, reliability, and availability of video. We have designed PERQS to tackle all of these challenges efficiently. With this contributory system, event querying can be performed across all the diverse sets of participating cameras, providing a more comprehensive surveillance coverage. PERQS is especially useful when an event or incident is spread across multiple locations or multiple camera feeds from different organizations provide complete coverage of an area.

We aim to preserve the existing CCTV recording and storage architecture as much as possible so that organizations can easily integrate our system into their existing CCTV deployments without making significant changes. Doing so minimizes disruption and ensures a smooth transition to our system.

We prioritized a distributed design, as some organizations may want to keep their raw video feeds private. To address this, we came up with a distributed storage and video analytics architecture that avoids the need for sharing raw video feeds with others. Participating organizations store and analyze their video feeds on their private servers, whether they are located locally or remotely. This distributed design eliminates the need for centralization by allowing multiple organizations to participate and share the workload. Another advantage of the distributed design is scalability, with multiple servers and storage nodes that can handle the processing and storage of large volumes of video data. This allows the system to scale up easily as more cameras and participants are added to the network.

CCTV cameras may not work correctly sometimes, or the feed may become corrupted. To address this problem, we introduce the concept of *query consensus* in which multiple cameras can collectively participate in the query processing phase. This increases the system's reliability and ensures the query results remain accurate even if one camera is not working correctly. For locations without cameras or a small number of cameras, we propose a novel consensus mechanism called *time-travel consensus*. In time-travel consensus, we use other nearby cameras to identify events in the past and at locations leading up to the location of the incident (i.e., travel back spatiotemporally) to participate in the query result consensus. These cameras participate in achieving consensus about a past event that may be causally related to the event for which more evidence was needed.

The extra capability we require from each participant is to perform video analysis on their video feed. They set up their own ML server locally or use remote servers to upload the video for analysis. Furthermore, we have designed our system to be flexible and adaptable to different CCTV architecture options. Includes support for local storage, which allows recordings to be stored on-site; cloud storage, which allows for remote access and management of recordings; and intelligent cameras, which incorporate advanced features such as motion detection and facial recognition.

To summarize, we aimed to create a distributed system to handle unreliable participants, whether due to technical faults or malicious behavior, with minimal change. To meet these requirements, we decided to implement a blockchain-based architecture. We implemented mechanisms for video verification and query execution over all participants without sharing the raw video feed. By committing a hash of the video feed to the blockchain while recording, we can verify if the feed has been tampered with or corrupted at a later point in time. The query execution and consensus mechanisms can be easily integrated into the blockchain structure.

We chose to use a permissioned blockchain as we envision the system being limited to CCTV owners and official authorities. In a permissioned blockchain ecosystem, the participants who add entries to the shared log are not anonymous entities but authenticated entities whose identities are known to all other participants. With different roles and permissions, a permissioned blockchain is a more suitable design choice to facilitate the needs of these different groups of participants. Using a blockchain also makes the system

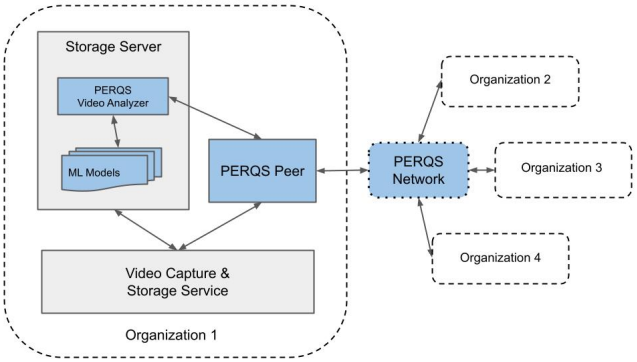

Figure 2: PERQS components in video capturing organizations.

fault-tolerant, as it is designed to be distributed and can withstand individual failures.

Overall, the design of PERQS aims to provide an efficient, scalable, and user-friendly permissioned system for querying and analyzing video footage from many CCTV cameras while ensuring video privacy. In the upcoming sections, we will explain the PERQS architecture and query execution in detail.

3.1 PERQS Architecture

The design of PERQS uses a blockchain-based architecture composed of several vital components, as shown in Figure 2. At the system's core are the participating CCTV owner organizations, individuals, or groups that own one or more CCTV cameras. These organizations are connected to the PERQS network and deploy specific PERQS components alongside their CCTV video capture and storage service. The main components are:

- **PERQS Network:** A permissioned blockchain network connects all the participating CCTV owner organizations in PERQS. Access to the network is restricted to verified members, ensuring that only actual CCTV owner organizations can participate. Each organization has one or more blockchain peers for participation in the network, allowing for efficient and secure communication.
- **PERQS Client:** It is the blockchain peer that enables the participation of organizations in the PERQS network. It has multiple roles, such as committing the video hash to the blockchain, decoding and sending queries for execution, and reaching a consensus on query results. These functionalities are explained in more detail in the upcoming sections of the paper.
- **Video Capture & Storage Service:** In an organization, CCTV cameras are managed and stored using a central system called the video capture and storage service. Depending on the organization's preference, storage service can be on-premises or remotely. One of the critical features of PERQS is that it does not require any changes to the existing recording systems, but it must be integrated with the PERQS video analyzer.
- **PERQS Video Analyzer:** Responsible for performing video analysis on the recorded videos. The queries are passed to the video analyzer, which decodes and executes them on the individual video feeds. The organizations set up their video analysis servers locally or remotely. The video analyzer is designed to be extensible, so it can easily be updated with new video analytics models for improved accuracy. This allows the system to stay up-to-date with the latest advancements in video analysis technology, ensuring that the system remains effective in providing accurate results.

To summarize, PERQS is a decentralized system that enables organizations to collaborate in analyzing video footage without sharing their raw video feeds. Organizations must be able to conduct video analysis on their feeds using machine learning models. Participants are linked together using a blockchain network, and video committing ensures the integrity of the recordings, making them tamper-proof.

3.2 Public Event Recording and Committing

In PERQS, CCTV cameras record the video footage, which is then saved to the affiliated organization's servers (i.e., the organization that owns the CCTV cameras). The PERQS client, integrated with the organization's video capture and storage service, computes a video hash and commits it to the blockchain. This process, in effect, stores a commitment of the captured video feed onto the blockchain and allows the system to verify later if the video feed has been tampered with or corrupted by the video feed owner.

PERQS requires minimal changes to the current CCTV video recording system. The process involves saving video streams into a file at certain intervals, computing a hash summary, and sending it for commitment. The interval at which the videos are saved can be configured for optimal video analyzer performance. Apart from this step of interval saving and hash computation, all other parts of the existing CCTV recording system remain unchanged. This allows organizations to easily integrate PERQS into their existing video capture and storage infrastructure.

3.3 Event Querying

PERQS allows only select participants, such as public authorities and CCTV owners, to send queries and receive results. It utilizes a query language similar to SQL- PERQL(Public Event Recording and Querying Language), which allows users to specify a time and location of interest. This information is used to determine which cameras are likely to have captured the event in question. The query is then passed on to the camera owners, who perform video analytics on their saved video feeds and send the results back to the network. By combining the results from multiple organizations, PERQS forms a consensus result. This *query consensus* provides a comprehensive and accurate view of the event in the query. This consensus-based approach ensures that the system is reliable and fault-tolerant, even if one or more cameras are not working correctly or their video feeds are corrupted. In Section 4, we will provide a detailed explanation of the query execution engine in PERQS.

PERQS requires participants to register the cameras with their GPS location and field of view directions. This information narrows down the cameras of interest while performing a query execution. We use two oracles *1)* camera finder and *2)* timing oracle. An *oracle* is a trusted external entity that can provide off-chain data to the blockchain network[15]. Oracles can fetch data from external sources such as APIs or databases and make it available to the smart

contracts running on the blockchain network. As the name suggests, a *camera finder oracle* is a tool that locates cameras in a specific geographic area based on GPS coordinates. For example, if a user inputs the GPS coordinates of a location, the camera finder oracle will return information about all the cameras that are close to that location. Oracle can use any technique to find a set of interesting cameras. In our case, it uses Euclidean distance and a more complex search circle expansion[31].

On the other hand, a timing oracle like [42] is used to synchronize the clocks of multiple cameras. It is instrumental in a consensus setting, where multiple cameras may need to be time-synced to ensure accurate data collection and processing. When asked about two specific cameras, the timing oracle will return the time offset between the two cameras, which can then be used to adjust the clocks as needed. In our case, it is implemented using basic Network Time Protocol (NTP) [37] exchange between cameras.

4 QUERY EXECUTION ENGINE

The query execution is responsible for user query handling. It consists of several components: query language, decoding and execution, and video analyzers. The primary means of communication between the user and the system is through the Public Event Recording and Querying Language (PERQL), which allows users to prepare and send queries to the PERQS client. The query decoding and execution process is a collective effort of the PERQS client and the PERQS video analyzer. The client identifies the relevant videos and requests the analyzer to perform video analysis to generate query results. The results from the Video Analyzers combined to produce the final results.

4.1 Query Language - PERQL

PERQL is a new language that provides more flexibility and customization for querying camera feeds. This SQL-like language allows users to design their queries rather than being limited to pre-defined popular queries. We have extended FRAMEQL [28] for PERQS to support GPS location, time interval, and consensus mechanism. PERQL assumes that the output from the video analyzer is organized in a table format, and the query is run through this table to retrieve the desired information. This approach allows for a wide variety of queries to be executed, making it more versatile and efficient for handling large amounts of data. Key operations supported in PERQL and their syntax.

• **CREATE MODEL**

```
CREATE MODEL <model_name>
(
<data_field1> <data_type>,
<data_field2> <data_type>,
...)
```

An ML (Machine Learning) or vision model for video analysis is a trained algorithm that can detect and identify objects, people, or actions within a video. When a new machine learning or vision model is added to PERQS, it is distributed throughout the network using a query mechanism. This process allows organizations to update their servers with the new video analytic algorithms, ensuring that the system stays up-to-date with the latest advancements in video analysis technology. The *data_field*s are the output fields generated by these models.

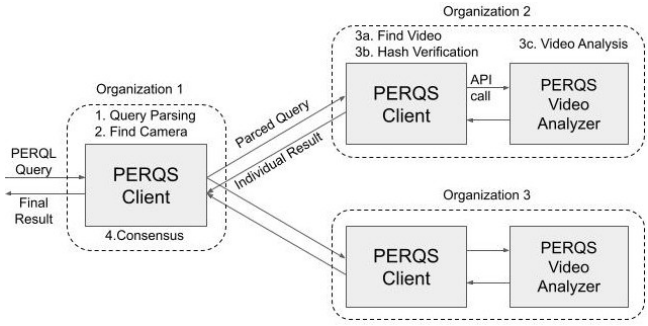

Figure 3: PERQL query execution stages. Organization 1 initiates a query execution. Send the parsed query to two owner organizations, 2 & 3.

• **ALTER MODEL**

```
ALTER MODEL <model_name>
ADD <data_filed> <data_type>
```

When there is a change in existing models, this query propagates it to all the organizations. Modifications to the ML model can result in additional data fields. For example, the *vehicleDetector* model can detect *vehicleType* and *vehicleModel*. If *vehicleColor* is added as a new field, it will require an ALTER MODEL query to distribute the changes across all organizations.

• **SELECT**

```
SELECT <fields>
FROM <model_name>
WITH <feature_parameters> -- optional
AT <gps_loc>
WHERE <conditions> -- optional
TIME <start_time> TO <end_time>
```

A SELECT query is used to extract information from the collective video database. The video feeds are narrowed down using the GPS location and time data. On the video feeds, the owner organization's video analyzer server applies an ML/vision model to perform analysis and create a table with the results. The query conditions are then used to filter out unwanted results. Using the WITH keyword, we pass optional parameters to any models if required. Finally, the results from multiple cameras are combined to form the final output of the SELECT query.

Using a query language in PERQS makes the system highly expandable, providing numerous possibilities for extracting information from the video database. The model plug-and-play design ensures that new updates and techniques in the field of computer vision can be quickly integrated into the system. The language used in the system, similar to SQL, makes it easy for developers to design queries for natural language requests. PERQL provides a more intuitive, user-friendly user experience and powerful querying capabilities. Make the system more accessible and efficient in providing accurate information from the video footage.

Table 2: An example consensus policy for different data fields.

	Output Type	Consensus Rules	Error handling/policies
1	String	Equals	Majority, at least *n*
2	Set of strings	Set Intersection / Set Union	Strings present with majority or *n* of the results
3	Range (time, integer, float)	Overlapping range	Floor or ceil of overlapping
4	Number(float/int)	Equals	±5% is considered as equal (x%)
5	Set of numbers(float/int)	Set Intersection	majority, ±5% is considered as equal
6	Color (hex string)	Equals	±5% in each RGB is considered as equal
7	Boolean(true/false, yes/no)	Equals	Majority, at least *n*

4.2 Query Decoding and Execution

Interpreting and executing queries is at the heart of the PERQS system. The entire execution engine is spread across clients, peers, and video analytic servers. When a query is initiated, it reaches the client and is forwarded to the organizations for video analysis. Finally, the results are sent back, and a consensus step is implemented to ensure that false results caused by malicious actors are eliminated. A single PERQS query execution consists of four stages, as shown in Figure 3.

① **Query parsing:** Decode the keywords and parameters from the query string. If the GPS location or time details are not in the query, we abort the execution.

② **Find the camera:** Identify the camera sets of interest based on the GPS location details. A *'camera-finder'* oracle returns the camera ID given GPS location and range. Each organization registers the camera location details to the *camera-finder* oracle at the time of joining or when a new camera is added. Once we identify the cameras, send the decoded query to each owner.

③ **Owner query execution:** CCTV owners find the corresponding video using the time parameter passed with the query. Then, the client verifies the hash of the video with the committed hash in the blockchain. If the hashes match, an API call is made to the video analytics server with the video, *video_id*, and decoded query parameters. It analyses the video and returns the result as a table per the query options. This result is sent back.

④ **Consensus:** Combine all the tables received using the consensus rules and policies. Example consensus rules for some widely used data types are in Table 2. New data types and policies will be added when new analyzers or ML models are added to the system. Policies include majority, at least n cameras, at least n organization. Default policies set by configuration can be overridden by query parameters if required. Instead of relying on the analysis of a single camera, the consensus algorithm requires a certain number of cameras, t, to participate in reaching a decision.

4.3 Video Analyzer

In PERQS, a video analyzer refers to a set of vision-based tools and modules to analyze video feeds and detect events, objects, or features. We consider the video analyzer module as a function that takes a video feed and specific parameters as input and returns a table as output. The columns of this table represent the output parameters of the model used, and each row represents an event or object that the vision model detected. This approach allows for easy and efficient processing of large amounts of video data, making it possible to extract valuable information from the feeds quickly. An example is an object Detector ML model, which will detect objects in each video frame.

4.4 Time-travel Consensus

PERQL query execution arrives at a consensus of an event happening when multiple cameras are available at a location, and the majority operate honestly. However, at certain locations, reaching a consensus may not be feasible because no cameras are present or the number of cameras that may have captured the event is not sufficient in number to reach a consensus. Alternatively, it may also be possible that even when cameras are present, the camera owners may behave maliciously or exhibit Byzantine behavior, leading to incorrect results, thereby thwarting consensus. *Time-travel consensus* proposes a solution to mitigate this limitation in certain cases.

Time-travel consensus can be likened to the ability to "go back and forth in time and space" to trace events that may have happened elsewhere geographically but are causally linked to the event of interest. To illustrate this, consider the example in Figure 4, which shows an accident that has occurred at a camera blind region. Even though the accident is not captured by any of the cameras, we can travel back in time to other locations where these two vehicles are captured by some CCTV cameras. The footage from these locations may be used to establish factors such as the speed of the vehicle or even its presence at the intersection where the accident happened. Likewise, we can travel forward by several time units from the event to identify the vehicle at other locations, which is especially useful if the accident was a hit-and-run.

The fundamental concept behind time-travel consensus is to perform a spatiotemporal search on the footage of nearby CCTV cameras to identify activities related to the event-of-interest in the past or future. These additional events help to reach a consensus, even when direct visual evidence is limited or absent. Major aspects of time-travel consensus are as follows:

- **Spatiotemporal Context:** Time-travel consensus recognizes that events in the real world are often interconnected across time and space. Often, a single event may be causally related to activities at other locations and at different times.
- **Comprehensive Analysis:** The primary objective of time-travel consensus is to support and validate the occurrence of the initial event. The PERQS system conducts a meticulous search of events in the spatiotemporal vicinity of a particular location when that

location has insufficient or no camera coverage. By examining the immediate vicinity and other nearby cameras at different timeframes, PERQS tries to piece together the patterns and sequences of events that may be causally related to the initial incident. Hence, time-travel consensus provides comprehensive coverage by expanding the scope of analysis beyond a single camera's field of view.

We use time-travel consensus when the camera-finder oracle returns a limited or zero count of cameras at a specified GPS location. The minimum number of cameras required for consensus and the consensus policies are configured beforehand. When the minimum camera count criterion is not met, the PERQS peer initiates a search for additional cameras by expanding the search radius R, which is specified as a configuration parameter. Subsequently, the query is executed on these newly identified camera feeds with a time offset Δ, which is again specified as a configuration parameter (or can be computed as a function of R). The specific values for the searching radius R and the time offset Δ are determined based on the geographical context of the search area. For instance, a smaller search radius in city limits may yield numerous cameras likely to capture pertinent events. However, a larger radius is required for remote areas with fewer cameras. Furthermore, the parameter Δ can often be computed as a function of R, e.g., by considering typical time intervals that a vehicle may have taken to travel the distance R. Approximate estimations suffice since the search occurs within a time interval rather than a fixed time. These R and Δ are established and configured through rigorous testing during the system's deployment phase. The value of R is expanded iteratively if enough cameras are not found.

After identifying the potential cameras and establishing an approximate time and radius Δ and R, respectively, the query is dispatched to initiate video analytics within the estimated time interval. Once the PERQS peer has received the results from all cameras, a search is conducted to identify related events that can substantiate the consensus regarding the event-of-interest. The endorsement of multiple supporting cameras is pivotal in reaching a final consensus. The effectiveness and precision of the time-travel consensus significantly depend upon the machine learning and vision models employed. These models must possess the capability to detect related events that provide corroborative evidence for the occurrence of the event-of-interest. We treat these models as black boxes in a plug-and-play manner. As advancements in ML and vision technology continue to enhance their performance and accuracy, this is a relatively minor concern in the future.

We can see in detail how time-travel consensus is used in the context of the car accident example shown in Figure 4, where we reach a consensus on the speeds of involved vehicles. A truck from *'B'* and a car from *'M'* collide in a camera blind area between *'D'* and *'I'*. Figure 4b and Figure 4c shows the positions of vehicles at times $t - \Delta_1$ and $t - \Delta_2$, respectively. A speed estimation model is run at junction *'I'* in the past for the time interval $(t - \Delta_1) - \theta$ and $(t - \Delta_1) + \theta$ and in the future for the time interval $(t + \Delta_3) - \theta$ and $(t + \Delta_3) + \theta$. Where 2θ is the time interval. Similarly, we run the speed estimation model with additional cameras at locations *'D'*, *'M'* and *'B'*. With the help of these additional results, we can reach a consensus on the speed of the vehicles. Suppose one vehicle was speeding over the allowed speed limit; we can suggest that

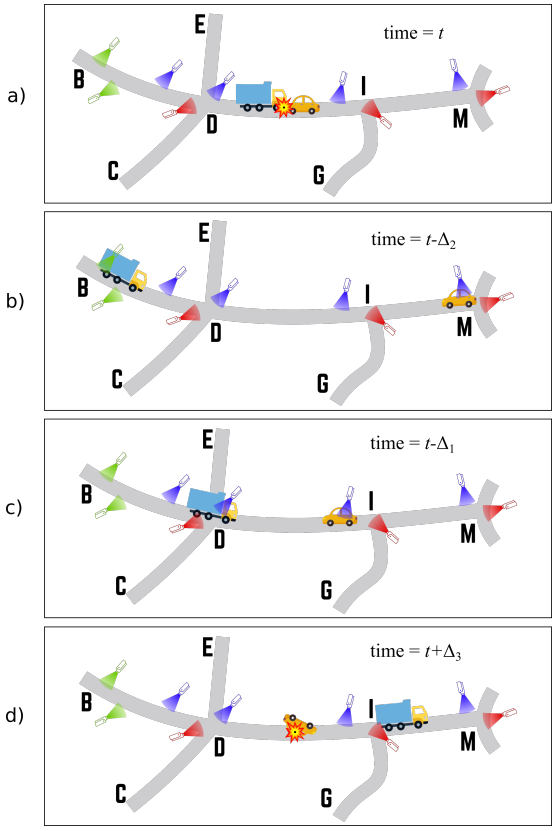

Figure 4: Example 2 Section 2 illustration for time travel consensus. a) An accident occurred. b) Positions of the vehicles at $time = t - \Delta_2$. c) Positions at $time = t - \Delta_1$. d) Hit and run. Position at $time = t + \Delta_3$. Only the cameras which might capture the vehicles are shown.

the accident might have occurred due to the overspeeding of that vehicle and catch the culprit.

Limitations: It is essential to note that the time-travel consensus mechanism will not always be able to produce results. For many events, supporting the occurrence with other activities observed by nearby cameras is not possible. Another limitation is the participation of irrelevant nearby cameras. From the event-of-interest, we search for cameras in a radius of R. It is possible that the majority of these cameras do not have any supporting events to participate in the consensus. Another issue is that time-travel consensus is not compatible with all video analytic models. Consider the car accident example in Figure 4. Say we have only an accident detection model. Since none of the cameras captured the accident, this model will not help to reach a consensus. We may only estimate the speed and direction of the vehicles with some other ML/vision models.

4.5 Multiple Queries

PERQL allows for the implementation of complex operations through the use of multiple queries. One example of such a complex operation is tracking a moving object across multiple CCTV cameras. In order to achieve this, queries would need to be sent to different

GPS locations at specific time offsets to track the movement of the object. PERQL supports these complex use cases and can be utilized by query experts to design the sequence of queries and logic required to combine the results. In our PERQS evaluation section, we demonstrated the use of PERQL to perform vehicle finding. However, these capabilities of PERQL are wider than just vehicle tracking and can be used to solve more complex applications with sufficient programming and SQL skills.

5 IMPLEMENTATION

Our PERQS prototype was built on Hyperledger Fabric v2.2 [22] and deployed on a Kubernetes cluster [32]. The video analytics servers were developed using the Python Django framework, utilizing Django v4.1.4, Python v3.10.6, and TensorFlow v2.8.0. In our implementation, the PERQS network is a Hyperledger Fabric network, with the Fabric client and Fabric peer performing the role of the PERQS client. Smart contracts were used to implement hash committing and query executions.

Hyperledger Fabric [6] is an open-source permissioned blockchain framework. it employs a unique architecture that separates the network's consensus process from the smart contract execution, enabling a higher degree of scalability and privacy. It can handle thousands of transactions per second and accommodate hundreds of organizations [16, 18]. Additionally, Hyperledger Fabric provides role-based access control, identity management, and confidentiality. Fabric also supports the use of smart contracts, which are written in a variety of programming languages such as Go, JavaScript, and Java.

5.1 PERQS Client

The Fabric client and Fabric peer work jointly to perform the role of the PERQS client. When the organization user sends a video hash or queries to the Fabric client, it utilizes the peer to commit the hash or execute the query. To facilitate the operation of the PERQS system, we have implemented two smart contracts within the Fabric peer. The first smart contract is responsible for hash commitment and the second is for query execution.

5.1.1 Video hash committer: To optimize the performance of our ML models, the video is saved in smaller chunks, with chunk size typically set to five minutes or less. When a chunk is saved, a hash is computed and a Fabric smart contract is executed. The Fabric client sends transaction proposals to the Fabric peer, which executes the smart contract and endorses the transactions. Fabric clients then send these endorsed transactions to the Fabric orderer to commit them to the blockchain. Once the committing process is complete, a new block with video hash is delivered to all peers.

5.1.2 Query execution: The PERQL queries are sent to the Fabric client for execution. The query is first decoded, and the GPS coordinates are identified. The Fabric client then sends the location information to the Camera Oracle, which returns the nearby camera IDs based on their GPS coordinates and direction of view. The Camera Oracle essentially acts as a database of active cameras.

Once the Fabric client has obtained the IDs of the nearest cameras, it prepares a transaction with the decoded query and sends it to the peers who own the nearest cameras. The peers then execute a smart contract with the decoded query. Within the smart contract, the corresponding video feeds are found and verified for hash integrity. From the smart contract, an API call is made to the video analysis server with the query and video details. The analytic server performs the video analysis and prepares an output table. This output is then sent back to the Fabric client along with the transaction response.

The Fabric client collects the results from all peers and performs a majority consensus or time-travel consensus, depending on the availability of the cameras, to determine the final query result.

Figure 5: CityFlowV2 [17]: Case study 1: time sync screenshots of 5 cameras when a blue SUV truck crossing the junction at 11:30:05.50.

6 EVALUATION

6.1 Experimental Setup

We set up a Fabric-based testbed for our experiments, consisting of three organizations. There is one orderer node, and each organization has a single peer node. Our Kubernetes cluster setup contains six nodes. All nodes run as a virtual machine and are allocated vCPUs of Intel(R) Core(TM) i9-7920X CPU, running at 2.90GHz. Fabric components are deployed as Kubernetes pods. We have set up our video analyzer as a python-django server, which utilizes GeForce GTX 1080 Ti GPU available in the server.

We have used the AICITY21 benchmark (CityFlowV2) [17], [14], captured by 46 cameras in a real-world traffic surveillance environment. A total of 880 vehicles are annotated in 6 different scenarios. There are 215.03 minutes of videos in total. We have identified eight junctions where we have multiple camera recordings.

6.2 Consensus

We conducted a simulation of a real-world scenario utilizing the CityFlowV2 dataset. Over the eight junctions, we try to reach a consensus of finding vehicle queries. On the PERQ video analyzer, we define a model *vehicleDetector* which uses YOLOv3 [48] frame by frame to detect the type of vehicles present (car, truck, bike, or SUV) in each frame. We used overlapping bounding boxes to combine the individual frame results. We then utilized the cropped images of the vehicles to determine their model and color. Our resulting machine learning model comprises a table structure that includes *vehicleID*, *timeAppeared*, *vehicleType*, *vehicleColor*, *vehicleModel*, and *featureVector*.

Table 3: Case study 1: reply from five cameras available at junction one and the final result after majority consensus.

timeAppeared	vehicleType	vehicleColor
Camera 1 result table		
11:30:04.90 11:30:07.10	truck	#012AB4
11:30:36.30 11:30:38.70	truck	#1336B8
11:32:08.40 11:32:09.90	truck	#15247F
11:33:12.60 11:33:14.50	truck	#3655AA
Camera 2 result table		
11:30:04.30 11:30:06.70	truck	#103BB8
11:30:35.90 11:30:37.70	truck	#1538B9
11:32:07.90 11:32:08.60	truck	#20247C
11:33:12.20 11:33:14.30	truck	#26456A
Camera 3 result table		
11:30:05.10 11:30:17.30	truck	#002CAD
11:30:36.50 11:30:48.70	truck	#1A35B4
11:32:08.20 11:32:19.30	truck	#25249B
11:33:12.90 11:33:23.20	truck	#36559A
Camera 4 result table		
11:30:01.80 11:30:06.80	truck	#003AB3
11:30:31.40 11:30:38.40	truck	#1254A4
11:33:09.20 11:33:14.10	truck	#36557A
Camera 5 result table		
11:30:04.80 11:30:09.20	truck	#001261
11:33:12.30 11:33:16.60	truck	#26356A
Final result table after consensus		
11:30:05.10 11:30:06.70	truck	#032CA2
11:30:36.50 11:30:37.70	truck	#1234B4
11:32:07.90 11:32:08.60	truck	#15245F
11:33:12.90 11:33:14.10	truck	#36558A

Case Study 1 - Majority consensus: Find a blue truck passed through junction one at around 11:30 AM. This case study is a simplified version of Example 1 described in Section 2 to demonstrate the majority consensus. Consider Figure 5; we have five views of the junction from five different cameras. Representing the query in our PERQL as follows

```
SELECT timeAppeared, vehicleType, vehicleColor
FROM vehicleDetector
AT 42.525678, -90.723601
WHERE vehicleType="truck", vehicleColor="#0000FF"
TIME 2023-01-25 11:30:00.005 TO 2023-01-25 11:35:00.005
```

① The query planning stage. Find the cameras available at junction one(42.525678, -90.723601) using the *camera-finder* oracle.

② Forward the query to each of the camera owners.

③ Each camera owner executes the query.

- From the time information, identify which stored video feed got the visuals
- Verify the hash commitment to check whether the video has been tampered with. In case of hash mismatch, abort.
- Call the *vehicleDetector* video analyzer with the video feed and fields required. Object detector returns a table with *vehicleID*, *timeAppeared*, *vehicleType*, *vehicleColor*, *vehicleModel*, and *featureVector*. Execute the condition to filter out rows not required. *vehicleType* and color similarity in this case. Filtered tables for these five cameras is shown in Table 3. Send the table back to the querier.

④ Collect the results from all camera owners. Use the time sync oracle to synchronize the tables. Then, compare the tables to construct the final table. If the majority of the cameras spotted a car, it would be there in the final list. The Table 3 shows the query execution results.

Figure 6: Case study 2: cycle captured by camera 10, 11, 12, 13, 15 for the find cycle query at different times.

Figure 7: Case study 2: all the three cycles captured by camera 15 at 10:01:06.10.

Case Study 2 - Time-travel consensus: Find a cyclist passing through junction eight. PERQL query is as follows.

```
SELECT timeAppeared, vehicleType
FROM vehicleDetector
AT 42.525678, -90.723601
WHERE vehicleType="cycle"
TIME 2023-01-25 10:00:00.000 TO 2023-01-25 10:05:00.000
```

The execution process is similar to Case Study 1, except that junction eight lacks overlapping camera views. Only two of the six

Table 4: Case study 2: final result after time-travel consensus.

timeAppeared	vehicleType
10:00:14.20 10:01:13.10	cycle
10:01:02.30 10:01:45.40	cycle
10:01:04.50 10:01:42.30	cycle

nearby cameras capture the junction's view. Therefore, we utilize time-travel consensus to merge the outcomes from the six cameras. One camera missed the cycles entirely because cycles took a turn before entering the field of view. The remaining cameras record the cycle, some a few seconds prior and another a few seconds after the junction. By merging the outcomes from these five cameras, we achieved a majority, hence the outcome. Steps 2 and 3 remain unchanged, but in step 4, we utilize time-travel consensus instead of a simple majority. The result is shown in Table 4.

Table 5: Consensus execution results summary on eight junctions.

J	Cameras IDs	Find Query	#Out	Consensus Type
1	1,2,3,4,5	Blue Truck	4	Simple Majority
2	6,7,8,9	Green Car	1	Simple Majority
3	36,38,39,40	White Car	2	Simple Majority
4	34,35,36,37	White SUV	1	Time-travel
5	29,30,31,32	Blue Car	2	Time-travel
6	22,23,24,25,26	White Truck	1	Time-travel
7	18,19,20,21	Silver Car	1	Time-travel
8	10,11,12,13,14,15	Cycle	3	Time-travel

Consensus Study Summary: Table 5 summarises the consensus experiment on eight junctions from the CityFlowV2 dataset [17]. The first three junctions had a majority of the available cameras pointing towards the junction, which allowed for the capture of the passing vehicles simultaneously. However, the cameras were not completely overlapping for the remaining five junctions. Thus, we used time-travel consensus to locate the vehicle's prior or post appearance using nearby cameras at a Δ time offset. We took this offset into account while merging the table results for consensus. The offset was determined based on the camera's distance from the junction and the average traffic speed on that road.

6.3 Query Execution Time

Query execution time can be broken down into query planning time, parsing time, video analytic model execution time, and consensus time. To measure query planning, query parsing, and consensus

Query Planning	Query Parsing	Video Analytics	Majority Consensus	Time-travel Consensus
31.5ms	230.5ms	1.72s	47.15ms	89.44ms

timings, we simulated the process 1000 times and calculated the average. For video analytics, we measured the time required for performing detection on our eight-junction case study. There were 36 videos, each averaging three minutes long and having a frame rate of 10 fps. For a three-minute video, if we process each frame with YOLOv3, it takes around four minutes, and combining the result and detecting color will also require around a minute. Therefore, on average, 1.72 seconds are required to process one second of video. The step that consumes the most time during query execution is the video analysis process. The impact of the other steps involved in the query execution is negligible.

As we do not have a real-world deployment of PERQS, we have not measured network latency. However, it will be in the order of milliseconds, which is unlikely to impact overall performance significantly.

6.4 Optimizations

The main bottleneck that hinders fast query execution is the time-consuming process of video analysis. To improve performance, we have implemented two optimizations. The first one involves parallelizing the object detection process. Instead of processing frames one by one, we utilized the power of the GPU to process multiple frames simultaneously, which resulted in a significant improvement in the speed of video analysis. The 3-minute video object detection will be finished in less than 1 minute. We can improve performance further by fine-tuning the parameters and using more efficient models.

The second optimization we implemented is caching video analytic results in the server. By storing the results of previous analyses in a cache, we can avoid performing the complete video analysis again. Instead, we can filter out the relevant results from the cache using the query constraints provided by the user and send them back as the response to the query.

7 RELATED WORKS

PERQS is a collaborative video querying system built on recent computer vision models. We are going through some of the related literature.

Video Querying Systems: Numerous studies on video querying systems have focused on various aspects, including feature extraction, query interface design, and query optimization. For instance, Blazeit [28] proposed a system for efficiently performing spatiotemporal queries over large video datasets. They introduced FRAMEQL, a declarative extension of SQL for video analytics, which enabled optimized aggregation and cardinality-limited queries, resulting in up to 83x performance improvement. While [27] is a user-friendly end-to-end video query system.

In contrast, the QBIC system [19] was developed by IBM researchers in the 1990s for searching and retrieving images from large collections of visual data based on their content. Another study [8] discusses the implementation and evaluation of a video query processing system in the VDBMS Testbed, while [60], [7] focuses on the challenges of performing real-time video analytics on large-scale live video streams. Scanner [40] is a distributed system that enables users to perform various video analytics tasks on large-scale video data, such as object detection, tracking, and scene understanding. Miris [12] is a system designed to query object tracks in videos efficiently.

Additionally, VIVA [29] is an end-to-end system that allows users to interact with video data using natural language queries.

At the same time, PRIVID [13] is a practical privacy-preserving framework for video analytics queries. Finally, Vstore [58] is a data store designed to perform analytics on large videos.

Video Analysis: Several deep learning architectures have been developed for object detection, video classification, and action recognition in videos. YOLOv3 [41] uses a powerful backbone network based on Darknet-53 for real-time object detection, while Faster R-CNN [43] significantly improves the speed of object detection. 3D Convolutional Networks (3D CNNs) [47] process video frames as a 3D stacked frame volume, including temporal information for video analysis. Non-local Neural Networks [54] capture long-range dependencies in images and videos.

Convolutional Neural Networks (CNNs) are used in [30] for video classification, while [50] improves video classification accuracy using self-supervised learning for 3D CNNs. Anomaly detection in videos is tackled in [20] using self-supervised and multi-task learning. Two-stream Convolutional Networks [44], involving two separate CNNs, are used for action recognition in videos, with TS-BiRCNN [45] improving this architecture. [53] aims to address the challenge of modeling long-term temporal dependencies in videos while maintaining computational efficiency.

In addition to the mentioned systems, Visor [39] provides confidentiality for both the user's video stream and the machine learning models in the event of a compromised cloud platform. Vigil [61] is a real-time distributed wireless surveillance system that leverages edge computing capabilities. Chameleon [25] is a controller that dynamically selects optimal configurations for existing neural network-based video analytics pipelines. EdgeEye [35] is an edge computing framework specifically tailored for real-time intelligent video analytics applications. Reducto [34] is a system that adaptively adjusts filtering decisions based on time-varying correlations in the video data. CrossRoI [21] utilizes the inherent physical correlations of cross-camera viewing fields to enhance video analytics performance. Lastly, Spatula [23] enables scalable cross-camera analytics by leveraging edge compute boxes.

8 CONCLUSION AND FUTURE WORK

PERQS is a contributory CCTV network solution that utilizes blockchain technology to ensure the integrity of the video feed. It enables multiple organizations to pool their CCTV footage for public safety while maintaining data privacy and security.

Limitations: Although the PERQS system offers numerous advantages in terms of efficient querying and privacy of CCTV footage, it is essential to acknowledge its inherent limitations. One significant limitation is its inability to perform joint video comparisons effectively. Joint video analysis is crucial in various situations, including tracking the movement of objects or individuals across multiple camera views [55, 62], identifying patterns and correlations between different events, or conducting synchronized monitoring of specific areas [9, 10, 26, 33, 36, 57]. These tasks require the ability to compare and analyze videos concurrently to derive meaningful insights and make accurate judgments. With PERQS's current architecture, which does not involve the sharing of video streams, it is not possible to directly perform a joint analysis of the two video streams. Instead, separate queries would need to be executed for each location, retrieving and analyzing the footage individually.

Future Possibilities: We can introduce a secure hardware-based solution to handle multi-video analysis. Participants can send their private videos encrypted to the secure execution environment, and a joint analysis can be performed. This will require additional hardware support from participants.

To enhance the appeal of PERQS, we can add a user-friendly interactive UI that facilitates easy querying of the system. Another option is to include a natural language interpreter, which can convert questions into the PERQL query format. The backend of these can use our core engine to execute the queries. These additional features would make PERQS more accessible to more users.

ACKNOWLEDGMENTS

Grants from the Department of Science and Technology's National Mission on Inter-Disciplinary Cyber-Physical Systems (via IHUB-IIT Kanpur), the National Security Council (via Indian Urban Data Exchange), and the Indian Institute of Science funded our work.

REFERENCES

[1] 1211 2022. An overview of video analytics in security. https://www.ifsecglobal.com/video-surveillance/overview-video-analytics-security/ (accessed on 23 June 2023).

[2] 1231 2022. Surveillance camera statistics: which cities have the most CCTV cameras? https://www.comparitech.com/vpn-privacy/the-worlds-most-surveilled-cities/ (accessed on 10 March 2023).

[3] 1234 2021. Global Surveillance Camera Markets 2021-2025. https://www.prnewswire.com/news-releases/global-surveillance-camera-markets-2021-2025---integration-of-ai-systems-adoption-of-iot-based-surveillance-systems--spy-and-hidden-cameras-growth-in-transition-from-analog-to-ip-cameras-301359651.html (accessed on 10 March 2023).

[4] 1241 2021. Adoption of advanced video analytics is "rising rapidly", NW Security UK study finds. https://www.ifsecglobal.com/video-surveillance/adoption-of-advanced-video-analytics-is-rising-rapidly-nw-security-uk-study-finds/ (accessed on 10 March 2023).

[5] 1251 2021. INDIA CCTV MARKET - GROWTH, TRENDS, COVID-19 IMPACT, AND FORECASTS (2022 - 2027). https://www.mordorintelligence.com/industry-reports/india-cctv-market (accessed on 10 March 2023).

[6] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, and et al. 2018. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In *ACM Eurosys*.

[7] Guilherme H. Apostolo, Pablo Bauszat, Vinod Nigade, Henri E. Bal, and Lin Wang. 2022. Live Video Analytics as a Service. In *Proceedings of the 2nd European Workshop on Machine Learning and Systems* (Rennes, France) *(EuroMLSys '22)*. Association for Computing Machinery, New York, NY, USA, 37–44. https://doi.org/10.1145/3517207.3526973

[8] Walid Aref, Moustafa Hammad, Ann Christine Catlin, Ihab Ilyas, Thanaa Ghanem, Ahmed Elmagarmid, and Mirette Marzouk. 2003. Video Query Processing in the VDBMS Testbed for Video Database Research. In *Proceedings of the 1st ACM International Workshop on Multimedia Databases* (New Orleans, LA, USA) *(MMDB '03)*. Association for Computing Machinery, New York, NY, USA, 25–32. https://doi.org/10.1145/951676.951682

[9] Luca Ballan, Gabriel J. Brostow, Jens Puwein, and Marc Pollefeys. 2010. Unstructured Video-Based Rendering: Interactive Exploration of Casually Captured Videos. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)* (July 2010), 1–11.

[10] Aayush Bansal, Minh Vo, Yaser Sheikh, Deva Ramanan, and Srinivasa G. Narasimhan. 2020. 4D Visualization of Dynamic Events From Unconstrained Multi-View Videos. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 5365–5374. https://doi.org/10.1109/CVPR42600.2020.00541

[11] Richard Barker. 2022. How many CCTV cameras are in London? https://clarionuk.com/resources/how-many-cctv-cameras-are-in-london/ (accessed on 10 March 2023).

[12] Favyen Bastani, Songtao He, Arjun Balasingam, Karthik Gopalakrishnan, Mohammad Alizadeh, Hari Balakrishnan, Michael Cafarella, Tim Kraska, and Sam Madden. 2020. MIRIS: Fast Object Track Queries in Video. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 1907–1921. https://doi.org/10.1145/3318464.3389692

[13] Frank Cangialosi, Neil Agarwal, Venkat Arun, Srinivas Narayana, Anand Sarwate, and Ravi Netravali. 2022. Privid: Practical, Privacy-Preserving Video Analytics Queries. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Renton, WA, 209–228. https://www.usenix.org/conference/nsdi22/presentation/cangialosi

[14] AI CITY CHALLENGE. 2021. AI City Challenge Dataset 2021. https://www.aicitychallenge.org/2021-data-and-evaluation/ (accessed on 10 March 2023).

[15] Brian Curran. 2018. What are Oracles? Smart Contracts, Chainlink and The Oracle Problem. https://blockonomi.com/oracles-guide/ (accessed on 10 March 2023).

[16] Shivdeep Singh Dave Enyeart. 2023. Benchmarking Hyperledger Fabric 2.5 Performance. https://www.hyperledger.org/blog/2023/02/16/benchmarking-hyperledger-fabric-2-5-performance (accessed on 5 October 2023).

[17] Marta Fernandez, Paula Moral, Alvaro Garcia-Martin, and Jose M. Martinez. 2021. Vehicle Re-Identification Based on Ensembling Deep Learning Features Including a Synthetic Training Dataset, Orientation and Background Features, and Camera Verification.. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 4068–4076.

[18] Christopher Ferris. 2019. Does Hyperledger Fabric perform at scale? https://www.ibm.com/blog/does-hyperledger-fabric-perform-at-scale/ (accessed on 5 October 2023).

[19] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Qian Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. 1995. Query by image and video content: the QBIC system. *Computer* 28, 9 (1995), 23–32. https://doi.org/10.1109/2.410146

[20] Mariana-Iuliana Georgescu, Antonio Bărbălău, Radu Tudor Ionescu, Fahad Shahbaz Khan, Marius Claudiu Popescu, and Mubarak Shah. 2020. Anomaly Detection in Video via Self-Supervised and Multi-Task Learning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), 12737–12747.

[21] 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* (Istanbul, Turkey) *(MMSys '21)*. Association for Computing Machinery, New York, NY, USA, 186–199. https://doi.org/10.1145/3458305.3463381

[22] Hyperledger. 2023. Hyperledger Fabric - GitHub. https://github.com/hyperledger/fabric

[23] Samvit Jain, Xun Zhang, Yuhao Zhou, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Paramvir Bahl, and Joseph Gonzalez. 2020. Spatula: Efficient cross-camera video analytics on large camera networks. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*. 110–124. https://doi.org/10.1109/SEC50012.2020.00016

[24] Facundo Lezama Javier Couto. 2022. A Guide to Video Analytics: Applications and Opportunities. https://tryolabs.com/guides/video-analytics-guide (accessed on 23 June 2023).

[25] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: Scalable Adaptation of Video Analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 253–266. https://doi.org/10.1145/3230543.3230574

[26] Wei Jiang and Jinwei Gu. 2015. Video stitching with spatial-temporal content-preserving warping. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 42–48. https://doi.org/10.1109/CVPRW.2015.7301374

[27] Manu Joseph, Harsh Raj, Anubhav Yadav, and Aaryamann Sharma. 2022. AskYourDB: An end-to-end system for querying and visualizing relational databases using natural language. arXiv:2210.08532 [cs.DB]

[28] Daniel Kang, Peter Bailis, and Matei Zaharia. 2018. BlazeIt: optimizing declarative aggregation and limit queries for neural network-based video analytics. *arXiv preprint arXiv:1805.01046* (2018).

[29] Daniel Kang, Francisco Romero, Peter D. Bailis, Christos Kozyrakis, and Matei Zaharia. 2022. VIVA: An End-to-End System for Interactive Video Analytics. In *12th Conference on Innovative Data Systems Research, CIDR 2022, Chaminade, CA, USA, January 9-12, 2022*. www.cidrdb.org. https://www.cidrdb.org/cidr2022/papers/p75-kang.pdf

[30] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-Scale Video Classification with Convolutional Neural Networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), 1725–1732.

[31] A. Khochare, A. Krishnan, and Y. Simmhan. 2021. A Scalable Platform for Distributed Object Tracking Across a Many-Camera Network. *IEEE Transactions on Parallel Distributed Systems* 32, 06 (jun 2021), 1479–1493. https://doi.org/10.1109/TPDS.2021.3049450

[32] Kubernetes. 2023. Kubernetes: Production Grade Container Orchestration. https://kubernetes.io/

[33] Jingwen Li, Lei Huang, and Changping Liu. 2012. People Counting across Multiple Cameras for Intelligent Video Surveillance. In *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance*. 178–183. https://doi.org/10.1109/AVSS.2012.54

[34] 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* (Virtual Event, USA) *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 359–376. https://doi.org/10.1145/3387514.3405874

[35] 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* (Munich, Germany) *(EdgeSys'18)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3213344.3213345

[36] Sujit Biswas Milind Naphade, Vinay Kolar. 2018. Multi-Camera Large-Scale Intelligent Video Analytics with DeepStream SDK. https://developer.nvidia.com/blog/multi-camera-large-scale-iva-deepstream-sdk (accessed on 23 June 2023).

[37] D.L. Mills. 1991. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications* 39, 10 (1991), 1482–1493. https://doi.org/10.1109/26.103043

[38] Thomas L. Norman. 2017. Chapter 6 - Electronics Elements: A Detailed Discussion. In *Effective Physical Security (Fifth Edition)*, Lawrence J. Fennelly (Ed.). Butterworth-Heinemann, 95–137. https://doi.org/10.1016/B978-0-12-804462-9.00006-3

[39] Rishabh Poddar, Ganesh Ananthanarayanan, Srinath Setty, Stavros Volos, and Raluca Ada Popa. 2020. Visor: Privacy-Preserving Video Analytics as a Cloud Service. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 1039–1056. https://www.usenix.org/conference/usenixsecurity20/presentation/poddar

[40] Alex Poms, Will Crichton, Pat Hanrahan, and Kayvon Fatahalian. 2018. Scanner: Efficient Video Analysis at Scale. *ACM Trans. Graph.* 37, 4, Article 138 (jul 2018), 13 pages. https://doi.org/10.1145/3197517.3201394

[41] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. *ArXiv* abs/1804.02767 (2018).

[42] E. Regnath, N. Shivaraman, S. Shreejith, A. Easwaran, and S. Steinhorst. 2020. Blockchain, what time is it? Trustless Datetime Synchronization for IoT. In *2020 International Conference on Omni-layer Intelligent Systems (COINS)*. https://doi.org/10.1109/COINS49042.2020.9191420

[43] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (2015), 1137–1149.

[44] Karen Simonyan and Andrew Zisserman. 2014. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1* (Montreal, Canada) *(NIPS'14)*. Cambridge, MA, USA, 9 pages.

[45] Bharat Singh, Tim K. Marks, Michael J. Jones, Oncel Tuzel, and Ming Shao. 2016. A Multi-stream Bi-directional Recurrent Neural Network for Fine-Grained Action Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 1961–1970.

[46] Saleem Durai M. A. Sreenu G. 2019. Intelligent video surveillance: a review through deep learning techniques for crowd analysis. In *Journal of Big Data 6*. https://doi.org/10.1186/s40537-019-0212-5

[47] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. 2015. Learning Spatiotemporal Features with 3D Convolutional Networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA. https://doi.org/10.1109/ICCV.2015.510

[48] Ultralytics. 2023. YOLOv3 in PyTorch > ONNX > CoreML > TFLite. https://github.com/ultralytics/yolov3

[49] Mark Patel Vasanth Ganesan, Yubing Ji. 2016. Video meets the Internet of Things. https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/video-meets-the-internet-of-things (accessed on 23 June 2023).

[50] Duc-Quang Vu, Ngan T. H. Le, and Jia-Ching Wang. 2021. Self-Supervised Learning via multi-Transformation Classification for Action Recognition. *ArXiv* abs/2102.10378 (2021).

[51] Han Wang, Shangyu Xie, and Yuan Hong. 2020. VideoDP: A Flexible Platform for Video Analytics with Differential Privacy. *Proceedings on Privacy Enhancing Technologies* 2020 (2020), 277 – 296.

[52] Junjue Wang, Brandon Amos, Anupam Das, Padmanabhan Pillai, Norman Sadeh, and Mahadev Satyanarayanan. 2017. A Scalable and Privacy-Aware IoT Service for Live Video Analytics. In *Proceedings of the 8th ACM on Multimedia Systems Conference* (Taipei, Taiwan) *(MMSys'17)*. Association for Computing Machinery, New York, NY, USA, 38–49. https://doi.org/10.1145/3083187.3083192

[53] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. 2016. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In *European Conference on Computer Vision*.

[54] X. Wang, Ross B. Girshick, Abhinav Kumar Gupta, and Kaiming He. 2017. Non-local Neural Networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017).

[55] Xue Wang, Jianbo Shi, Hyun Soo Park, and Qing Wang. 2017. Motion-Based Temporal Alignment of Independently Moving Cameras. *IEEE Transactions on Circuits and Systems for Video Technology* 27, 11 (2017), 2344–2354. https://doi.org/10.1109/TCSVT.2016.2581659

[56] Danielle Whittaker. 2021. Why AI CCTV is the future of security and surveillance in public spaces. https://www.securitymagazine.com/articles/96719-why-ai-cctv-is-the-future-of-security-and-surveillance-in-public-spaces (accessed on 23 June 2023).

[57] Xinyi Wu, Zhenyao Wu, Yujun Zhang, Lili Ju, and Song Wang. 2019. Multi-Video Temporal Synchronization by Matching Pose Features of Shared Moving Subjects. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. 2729–2738. https://doi.org/10.1109/ICCVW.2019.00334

[58] Tiantu Xu, Luis Materon Botelho, and Felix Xiaozhu Lin. 2019. VStore: A Data Store for Analytics on Large Videos. In *Proceedings of the Fourteenth EuroSys Conference 2019* (Dresden, Germany) *(EuroSys '19)*. Association for Computing Machinery, New York, NY, USA, Article 16, 17 pages. https://doi.org/10.1145/3302424.3303971

[59] Xiaoyi Yu, Kenta Chinomi, Takashi Koshimizu, Naoko Nitta, Yoshimichi Ito, and Noboru Babaguchi. 2008. Privacy protecting visual processing for secure video surveillance. In *2008 15th IEEE International Conference on Image Processing*. 1672–1675. https://doi.org/10.1109/ICIP.2008.4712094

[60] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. 2017. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 377–392. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/zhang

[61] Tan Zhang, Aakanksha Chowdhery, Paramvir (Victor) Bahl, Kyle Jamieson, and Suman Banerjee. 2015. The Design and Implementation of a Wireless Video Surveillance System. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (Paris, France) *(MobiCom '15)*. Association for Computing Machinery, New York, NY, USA, 426–438. https://doi.org/10.1145/2789168.2790123

[62] Kang Zheng, Hao Guo, Xiaochuan Fan, Hongkai Yu, and Song Wang. 2016. Identifying Same Persons from Temporally Synchronized Videos Taken by Multiple Wearable Cameras. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 810–818. https://doi.org/10.1109/CVPRW.2016.106