VU: Video Usefulness and Its Application in Large-Scale Video Surveillance Systems: An Early Experience

Hui Sun
School of Computer Science and Technology, Anhui University
Hefei, Anhui, China
sunhui@ahu.edu.cn

Xu Liang
School of Computer Science and Technology, Anhui University
Hefei, Anhui, China
ahu_lx@163.com

Weisong Shi
Department of Computer Science, Wayne State University
Detroit, MI, USA
weisong@wayne.edu

ABSTRACT

In the era of smart and connected communities, a video surveillance system, which usually involves tens and thousands of video cameras, has increasingly become a prominent component for the public safety. In the current practice, when the video surveillance system has a failure, the operation and maintenance team usually spends a lot of time to identify and locate the failure, which cannot guarantee real-time in a large-scale video surveillance system. Meanwhile, the video data with a failure wastes amount of storage space in the cloud. The emergence of edge computing is very promising in the preprocessing for source video data at an edge camera, and video surveillance systems are one of the popular applications for edge computing. In this paper, we propose ***VU***, a *V*ideo *U*sefulness model for large-scale video surveillance systems, and explore its application, such as early failure detection and storage saving. The VU model evaluates the usefulness of video data in a real-time fashion and notifies failures to end-users on the fly.

This paper has three contributions: *(1) a comprehensive video usefulness model has been proposed. To the best of our knowledge, this is the first work aiming to quality the video usefulness in a real application; (2) real-time failure detection algorithms based on edge computing and cloud computing are proposed to efficiently improve the mean time to repair (i.e., MTTR); (3) effective storage and bandwidth saving schemes for large-scale video surveillance systems are proposed and implemented.*

Results from a university-wide surveillance system consisting of 2,960 cameras show that failures of video data in different domains are accurately detected by VU model. MTTR is largely shortened by the fast detection algorithm in real time. The video data with the worst degree of VU is mostly discarded to reduce overload in the network and save storage space in the cloud.

CCS CONCEPTS

•**Computing methodologies** →*Modeling methodologies;*

KEYWORDS

Video usefulness, edge computing, cloud computing, video surveillance systems

SmartIoT'17, San Jose, CA USA

DOI: 10.475/123_4

ACM Reference format:
Hui Sun, Xu Liang, and Weisong Shi. 2017. ***VU***: Video Usefulness and Its Application in Large-Scale Video Surveillance Systems: An Early Experience. In *Proceedings of Workshop on Smart Internet of Things, San Jose, CA USA, October 2017 (SmartIoT'17),* 6 pages.
DOI: 10.475/123_4

1 INTRODUCTION

With the expansion of the city scale, public safety is essential for urban stability [11]. Video surveillance systems [18] mostly provide video evidence and recognition for criminal and civil cases [6]. Video surveillance systems have been essential to public safety [16][11] monitoring for a large-scale city's ecosystem. A large city such as Beijing or London has about one million cameras deployed [8]. Video cameras capture huge amounts of video data in the manner of uninterruptedly operation for 7*24 hours. Intelligence approaches (e.g., moving object detection [9] or behavior analytic [19]) are proposed to process video data at back-end.

For a large-scale video surveillance system, there are two major challenges: 1) How to effectively use the large-scale video data? 2) How to manage large-scale video surveillance systems when a failure occurs in edge cameras, end-users, network, or cloud?

In current video surveillance systems, a majority of failures impacting on usefulness of video data cannot be accurately identified and located in real time. Staffs are employed to view these failures from video image. This method consumes a large amount of manpower and time rather than meets needs of real-time surveillance. A video camera with a failure can products useless video data which is uploaded to cloud and wastes storage space before the failure is detected. Currently, there is rarely effective method to handle the useless video data.

Our Vision: To solve above problems, we firstly propose a **V**ideo **U**sefulness detection model for large-scale video surveillance systems (VU for short) based on edge computing [17] and cloud computing [1]. Edge computing has emerged along with the development of applications [20][3][2] in the realm of IoT [13] and IoE [14]. Real-time video analytic in a video surveillance system [15][4] is one of the most prevalent use-cases for edge computing. By this model, a detection system is designed to efficiently identify the component state of video surveillance systems and locate its failure to shorten their mean time to repair (i.e., MTTR) [5]. Video data with the lowest VU (useless video data) will not be uploaded to cloud, which reduces burden on network bandwidths and improves storage utilization in the cloud. This paper has three contributions:

(1) a comprehensive video usefulness model has been proposed. To the best of our knowledge, this is the first work aiming to quality the video usefulness in a real application;

(2) real-time failure detection algorithms based on edge computing and cloud computing are proposed to efficiently improve the mean time to repair (i.e., MTTR);

(3) effective storage and bandwidth saving schemes for large-scale video surveillance systems are proposed and implemented.

The remainder of this paper is organized as follows. Section II describes motivation for VU model study. The VU model and its exploration in three failure domains are presented in Section III. Some use cases are analyzed in Section IV. Finally, we make a conclusion for this work.

2 MOTIVATION

There are rich literatures on quality of video services, such as quality of service (QoS) [7] and quality of experience (QoE) [12]. **QoS** focus on video quality from the perspective of video service providers. **QoE** includes manual interaction with the evaluation of video quality in cloud. These criteria hardly evaluate the usefulness of video data.

Experiment are carried out in the video surveillance system with 2,960 cameras at Anhui University. Results show that video data usually has QoS or QoE problem. We also find that the video data meets QoS metrics and is clearly displayed; however, the useless video data (e.g., obstacle problem) is continuously transmitted to cloud, which largely overloads network and wastes storage space, especially for high-resolution video (such as 4K) [10]. The operation and maintenance team spend a great deal of time locating failures of useless video.

We are motivated to propose a video usefulness model (VU for short) for large-scale video surveillance systems based on edge computing and cloud computing. By edge computing, video data is pre-processed and some failures can be located, which reduces the burdens on network and optimizes video data storage in the cloud. Deep-level usefulness evaluation can be carried out in the cloud. The useful video data is in real-time evaluated meanwhile the state of video system is monitored under unmanned situation.

3 *VIDEO USEFULNESS* MODEL IN VIDEO SURVEILLANCE SYSTEMS

3.1 VU Model

VU is the first model to characterize the usefulness of video data comparing with QoS or QoE. This model relates with edge computing and cloud computing. Edge computing pre-processes video data in cameras while cloud computing provides largely storage and computing-intensive services at back-end.

Storage space occupied by video data depends on its usefulness. In this paper, D_{e1} represents the amount of video data from a single camera will be stored in the cloud in the range of t ∈ $[0, N_s]$ seconds. The d_p indicates the performance of video data transmission (MB/s). During the interval between 0 and N_s seconds, D_{e1} can be expressed as

$$D_{e1} = \sum_{t=0}^{t=N_s} (d_p * t * U(t)) \tag{1}$$

where *U(t)* is the value of VU and can be referred as

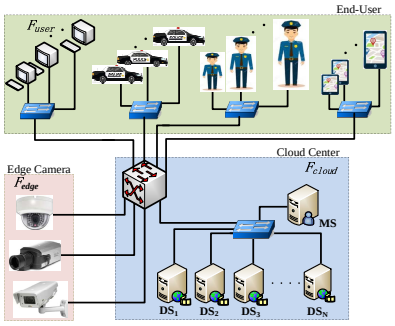


Figure 1: Failure Distribution in a Video Surveillance System.

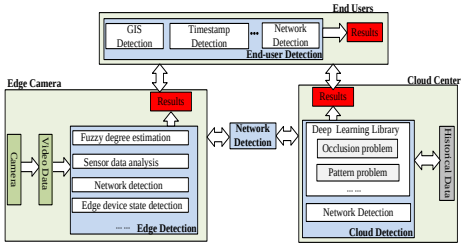


Figure 2: Evaluation Framework for VU model.

$$U(t) = \begin{cases} 1, & \Delta v > 0 \\ 0, & \Delta v \le 0 \end{cases} while \quad t \in (t_i, t_{i+1}) \tag{2}$$

where Δv is referred as the usefulness boundary of video data under scenarios, when $\Delta v > 0$, the video data is useful and uploaded to cloud. $\Delta v \le 0$, the video data is useless and has low VU values. To this problem, two methods are designed in this paper. Firstly, video data transmission from cameras is directly terminated according to values of VU. Secondly, its VU value is sent to end-users for further decision on video data.

3.2 Failure Domain in VU Model

Given experiment from the platform of video surveillance systems in Anhui University, we find that failures in this platform are mainly distributed in three scenarios defined as VU *failure domains* one of which has diverse types of failures for video data in the VU model.

We present failures in three VU domains (see Fig.1). In F_{edge} domain, the basic component is a camera. Police equipment and operation and maintenance team make up the F_{user} domain. Metadata services (MS for short) and data services (i.e., DS) make up the F_{cloud} domain. Failure in F_{edge} domain (see F_{edge_i} in Table 1), F_{user} domain (see F_{user_i} in Table 2), F_{cloud} domain (see F_{cloud_i} in Table 3) indicates a failure exists in an camera, an end-user, and a cloud service, respectively. Detail for a failure is described as followings.

3.3 Exploration of VU Evaluation Framework and Method

We provide the detail of evaluation framework and methods for VU in video surveillance systems.

3.3.1 VU Evaluation Framework. VU values for video data in a domain are measured based on video data content. Evaluation

Table 1: Failure in F_{edge} Domain.

F_{edge}	Failure Type	Description
F_{edge_1}	Camera offline	Edge camera without Internet connection.
F_{edge_2}	White screen	Full white screen with no images.
F_{edge_3}	Green screen	Full green screen with no images.
F_{edge_4}	Black screen	Full black screen with no images.
F_{edge_5}	Blurred screen	Fuzzy exists in blurring image with in screen.
F_{edge_6}	Pattern screen	Full pattern screen with images.
F_{edge_7}	Artificial shelter	Screen covered by an object, e.g., bags, daubing cover screen.
F_{edge_8}	Natural occlusion	Natural objects occlusion, e.g., cameras occluded by leaf.
F_{edge_9}	Abnormal characters	Abnormal characters in the image.
F_{edge_10}	Dark image	Screen with dark portion.
F_{edge_11}	Flicker screen	Image flickering in full screen.
F_{edge_12}	Rotary failure in pan/tilt	Pan/tilt in cameras cannot rotate.
F_{edge_13}	Self-test failure	Failure in power-on-self-test.
F_{edge_14}	Pre-position failure	Camera pre-position failure.

approaches employed in this model include background extraction, image contrast, and etc.

VU evaluation framework (see Fig.2) is composed of cameras, end-users, and cloud. In Fig.2, a part of failure-detection methods (e.g., fuzzy degree estimation) require edge computing to evaluate VU values. Results are sent to cloud and end-users, respectively. Edge detection handles useless video data using VU values and decisions are made by end-users.

Some failures (occlusion, pattern, and etc.) cannot be completely identified at the edge. A part of detection work can be pre-processed at the edge and intermediate results are sent to cloud, where the final results are obtained. Similar to edge, cloud handles video data with lower VU and then sends the decision to end-users. The VU degree for video data is fed back to the edge, which handles useless video data or notifies end-users.

Based on this framework, Failure detection schemes are designed to evaluate the degree of VU for video data.

3.3.2 VU Evaluation Method. By the above VU evaluation framework, experiments are carried out to detect different failures in three domains. It mentioned that these failures are classified accoring to large-scale video data from about 2,960 edge cameras.

(1) VU model of video data in F_{edge} Domain

There are fourteen failures (see Table I) in VU model for video data in F_{edge} domain. These failures are detected by four methods in this section.

Δ**Type-1** F_{edge_2}, F_{edge_3}, F_{edge_4}, F_{edge_5}:

F_{edge_2} (white screen), F_{edge_3} (green screen), and F_{edge_4} (black screen) make full screen without images into the white, green, and black mode, respectively. The colors of most pixels have high similarity to their neighbors, which are easily measured by the color histogram. By edge computing, color histogram method is applied to measure the degree of VU for video data with above failures. Useless video data is unnecessarily uploaded to cloud. F_{edge_5} (blurred screen) causes the video fuzzy with different levels, which is evaluated by the fuzzy degree estimation based on edge computing. When the VU degree is higher than reference ones, the camera stops sampling and notify end-users to handle the video data.

Δ**Type-2** F_{edge_6}, F_{edge_7}, F_{edge_8}, F_{edge_9}, F_{edge_10}, F_{edge_11}:

F_{edge_6} (pattern screen), F_{edge_7} (artificial shelter), F_{edge_8} (natural occlusion), and F_{edge_9} (abnormal characters) make full use of edge computing at an edge camera and cloud computing in the cloud. The former is employed in pre-processing and the latter is used in post-processing. At the edge, we apply background extraction method into pre-processing for video data based on edge computing. Intermediate results are uploaded to cloud and end results is obtained by means of deep learning library based on cloud computing. The detection system in cloud outputs VU values for video data with F_{edge_6}, F_{edge_7}, or F_{edge_8} failure. VU values lower than reference ones are sent to the detection system at edge to handle video data. While VU values near to reference ones are sent to end-users to make decision.

Video data with F_{edge_10} (dark image) or F_{edge_11} (flicker screen) is detected using deep learning library and results of VU are processed by edge cameras or end-users as the same methods mentioned above.

Δ**Type-3** F_{edge_12}, F_{edge_13}, F_{edge_14}:

F_{edge_12} (rotary failure in pan/tilt), F_{edge_13} (self-test failure), and F_{edge_14} (pre-position failure), relate to the camera state. This state detection is configured in the software stack at cameras. If one of these failures occurs, the detection system captures it and notifies end-users. Video data with F_{edge_13} and F_{edge_14} cannot be uploaded to cloud, which alleviates network burden and improves storage in the cloud.

Last but not the least, F_{edge_1} (edge camera offline) is the one of the most important failures. The evaluation method for this failure is implemented in a detection system in the cloud or end-users. There are two failures in network connection, i.e., disconnection and problematic connection. Although feedback of TCP/IP data package is a basic method, the detection processing varied with different failures.

For network disconnection, the detection system in the cloud or at end-user sends a data packet to a camera and waits for a response packet. When response overtime is detected, network disconnection is reported to end-user. In problematic connection, when the response data package arrives, the serial number of which is compared with that in request one. When the serial number of the packet in response packet is zero or has the same value to request ones, the network with the edge has a problem connection. Finally, the prompt for F_{edge_1} is sent to end-users as well.

Table 2: Failure in F_{user} Domain.

F_{user}	Failure Type	Description
F_{user_1}	End-user offline	End-user without Internet connection.
F_{user_2}	Image lagging	Latency in data transmission between cameras and end-users.
F_{user_3}	GIS mark failure	Failures in GIS (Geographic Information System) for an camera.
F_{user_4}	Timestamp failure	Time stamp in video image mismatches that in operating system.

(2) VU model of video data in F_{user} Domain

For end-users, there are four failures (see Table II). For F_{user_3} and F_{user_4}, an image is sampled while the current timestamp in the operating system is recorded at once. By methods of character recognition and extraction on the image, GIS and timestamp are obtained from the current image. F_{user_4} is displayed and is sent to end-users when the timestamp on the detected image is inconsistent with that in operating system. The local detection system sends a request including this camera's IP address (position encoding value) to that in the cloud. This detection finds GIS from a database using

Table 3: Failure in F_{cloud} Domain.

F_{cloud}	Failure Type	Description
F_{cloud_1}	Cloud server offline	Cloud server without Internet connection.
F_{cloud_2}	Video lagging	Latency in data transmission between cameras and cloud.
F_{cloud_3}	Video lost	Video data cannot store in cloud.

the receiving IP address. The item is considered as (IP (key)|Position encoding value (key)|Address|...) in this database. Appropriate GIS is returned to the detection system in user end.

For F_{user_2} (video lagging), this failure is caused by the network latency, which can be detected according to the time interval between the request data packet being sent to an edge camera (or cloud) and receiving the response data packet. In addition, the metric of packet loss ratio, which is used in QoS to evaluate real-time video data, is calculated according to the average number of response data packets and latency.

For F_{user_1} (end-user offline), this failure is similar to those in F_{edge_1} (such as disconnection and problematic connection) by means of feedback of TCP/IP data package. Please refer to the F_{edge_1} in Type-A for more details.

(3) VU model of video data in F_{cloud} Domain

These failures (see Table III) happen in the cloud. In a video surveillance system, cloud stores video data and provides history video query services for special affairs. There are so huge amounts of video data in the cloud that man-management is extremely inefficient. A simple failure of video data is hardly found and notified to end-users in real-time. Then video data with low VU values (uselessness) from failure is continually stored in the cloud. Afterward, the service of historical video data can be seriously affected.

F_{cloud_1} exists in two kinds of network connection, i.e., from a camera to cloud and from an end-user to cloud. These detections are configured in edge-device level and end-user level, respectively. F_{cloud_1} (i.e., cloud server offline) has two types of failures (i.e., disconnection and problematic connection) are similar to those in F_{edge_1} or F_{user_1} using feedback of TCP/IP data package. We use the detection system at an edge camera as an example to illustrate about F_{cloud_1}. Please refer to the F_{edge_1} in Type-A for more details.

F_{cloud_2} (video data lagging) is much difficult to be detected than that in F_{user_2} (image lagging). Because the video data in F_{user} domain is sent to users rather than stored in the cloud. Video data with lagging failure is found when end-users searched it again, which is accepted in criminal cases. Therefore, we designed a system in the cloud to check network latency which is expressed as the duration between sending a request packet to a camera and the point of receiving the response packet. The metric of a packet loss ratio is used to measure QoS for video data. VU for video data is lower when the value is smaller than reference one. F_{cloud_3} (video lost) is incurred by packet loss.

3.4 Discussion

The proposed VU model was derived and inspired by large-scale video surveillance system; however, the proposed model can be easily adjusted for other video systems, given that the three domains are very generic.

4 EXPERIMENTS

To verify the credibility of the VU model, we carried out experimental explorations for the video surveillance system with 2,960 edge cameras at Anhui University. Among these cameras, there are 2,734 edge cameras just being on-line and 226 video cameras cannot absolutely work because of failures. Otherwise, the video data from these online 2,734 video edges still have usefulness, which can be detected by our VU model.

4.1 Experiment Setup

Using the framework in Fig.2, we set up an evaluation system to measure the VU values for video data in the video surveillance system at Anhui University. In the cloud, we employ Microsoft Cognitive Services with powerful cloud computing resources as cloud-based data processing in detection systems. We configure an edge computing server to simulate the data processing at cameras.

4.2 Use Cases

Five use cases are evaluated by the VU model. From experiments, failures in different domains are detected and the degree of VU are sent to end-users in real time. The mean time to repair (i.e., MTTR) is largely shortened. Meanwhile, the video data with a low VU values is unnecessary uploaded to the cloud, which saves storage space for useful video data.

Case 1: F_{edge_3} (green screen)

Algorithm 1 Green screen detection.

```
1:  procedure HISTOGRAM
2:    Dim img As Image
3:    Dim max, maxs, sum As FLOAT
4:    Dim h_bin, s_bin As INTEGER
5:    Upload image img
6:    Find img's maximum pixel max
7:    Find the point of max ( h_bin,s_bin )
8:     SUM =the sum of all the pixel of img
9:     MAXS =the sum of around the point of max 16 points
10:   if MAXS /SUM > 0.7 then
11:         Output ("Dominant color percentage MAXS /SUM")
12:         Output ("Full Screen with same color,and Full-Screen Error! ")
13:         StorageVideo(NO_SEND2CLOUD, CAMERA_IP)
14:   end if
15: end procedure
```

We select green screen failure (see Fig.3(a)) in a camera. The color histogram method is applied in the detection system using edge computing simulated at an edge server. Results of VU indicated by domain color percentage in the video can be measured by the color histogram method. Video data with failures cannot be uploaded to cloud. Then it alleviates overload on network and achieves storage saving in the cloud. The pseudo-code is presented in Algorithm 1.

It is noted that **StorageVideo()** is employed to send the video data or VU values with failure to cloud or end-users.

Case 2: F_{edge_5} (blurred screen)

Video data with blur (see Fig.3(b)) is also detected by the detection system implemented at the edge. Results are sent to end-users. The pseudo-code is listed in Algorithm 2.

Case 3: F_{edge_8} (natural occlusion)

This failure is handled by the collaboration of edge computing and cloud computing. Video data with occlusion (see Fig.3(c)) can be detected by background extraction at the edge and similarity

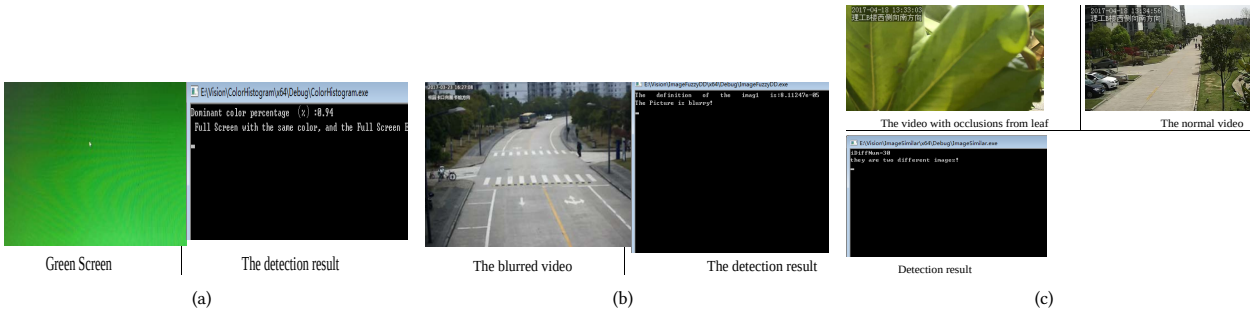


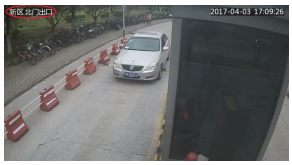
Figure 3: Failures detection in VU model taking F_{edge_3} (Green screen), F_{edge_5} (Blurred screen), F_{edge_8} (Natural occlusion) for example.

Algorithm 2 Blurred screen detection.

```
1: procedure IsFuzzy
2:    Dim image As Image
3:    Dim FD As FLOAT
4:    Upload image
5:    Compute image fuzzy degree FD
6:    if FD > 0.000131 then Output ("The picture is blury! ")
7:    else    Output ("The picture is not blury! ")
8:    end if
9:          StorageVideo(SEND2CLOUD, CAMERA_IP)
10: end procedure
```

Algorithm 3 Natural occlusion detection.

```
1: procedure ImageSimilarity
2:    Dim img, img1 As Image
3:    Dim idg As FLOAT
4:    Upload video test.mp4
5:    Upload correct image img1
6:    Extract background img1 from test.mp4 using edge computing
7:    Send img1 to cloud detection
8:    Compare the similarity of two images based on pixel using cloud computing
9:    idg =the similarity of two image from cloud detection
10:   if idg > 15 then Output ("A portion screen has occlusion! ") fiffiffi
11:   end if
12:   if idg >= 30 then Output ("The screen has full occlusion!")
13:   end if
14:         StorageVideo(NO_SEND2CLOUD, CAMERA_IP)
15: end procedure
```



GIS("新区北门出口") is error and the right one is "新区西门出口"

Figure 4: F_{user_3} (GIS mark failure).

comparison methods in the cloud. Results of VU for video data are fed back to edge. If the value of VU for video data is lower than the threshold, the detection system stops uploading video data and sends warning to end-users. Storage utilization in the cloud is optimized and the failure is located in real-time to shorten MTTR (see Algorithm 3).

Case 4: F_{user_3} (GIS mark failure)

In this case, we measure VU values for the video data with F_{user_3} failure at end-users (see Fig.4). This failure is sent to end-users to modify it. The edge camera sends a request for GIS based on its IP to a detection system in the cloud. GIS, which is found

Algorithm 4 GIS mark failure detection.

```
1: procedure GIS
2:    Dim img As Image
3:    Dim loc, loc1 As STRING
4:    Capture Image img from camera
5:    Recognize location (loc) from img and record the IP address of the camera
6:    Notify the camera's IP to cloud detection system
7:    Acquire the correct location (loc1) by the camera's IP in cloud
8:    if loc ! =loc1 then Output ("The camera's location is incorrect! ")
9:    end if
10:   StorageVideo(SEND2CLOUD, CAMERA_IP)
11: end procedure
```

by the detection in the cloud using IP address, is fed back to the camera. Comparing GIS with that in image, F_{user_4} failure can be obtained (see Algorithm 4).

Case 5: F_{edge_1} (edge camera offline)

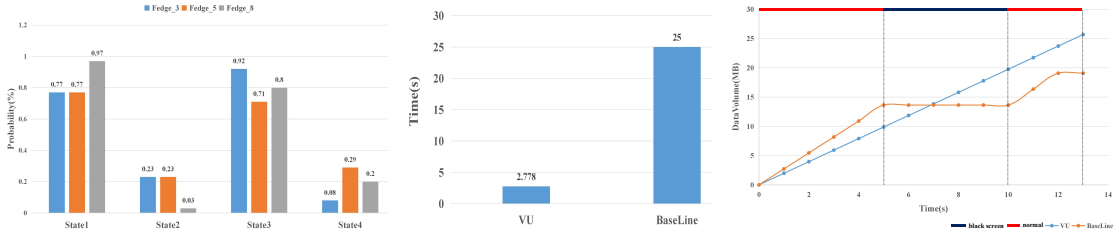
Algorithm 5 Edge camera offline detection.

```
1: procedure NetworkOffline
2:    Dim m As FLOAT
3:    m = Function(Send a test data packet to the detected camera)
4:    Then return the value of packet loss ratio
5:    if m ==0 then // response over time
6:          Output ("The network is disconnect! ")
7:    else(More incoming response packets of has the same serial)
8:          Output("the network has problematic connection! ")
9:    end if
10:   StorageVideo(SEND2CLOUD, CAMERA_IP)
11: end procedure
```

F_{edge_1} failure is detected by a detection system in cloud or end-users. Take a network failure in cloud for example, the detection system sends a test data packet to a camera and waits for the response packet. When the response time is found to be overtime, the network is regarded as disconnection. The detection system receives response data packets; however, these serial numbers in packets have the same value meaning that the network connection is problematic. Finally, the message of F_{edge_1} is sent to end-users as well (see Algorithm 5.)

5 PRELIMINARY RESULTS

Use cases above indicate that VU model provides real-time detection for failures in cameras, end-users, or cloud. Is VU model better or worse, e.g., its accuracy, performance, and storage saving? In this section, we present the accuracy, performance, and storage saving of methods in VU model based on a part of failures in the video surveillance system.



(a) (b) (c)

Figure 5: Accuracy, Performance, and Storage saving of failure detection in VU model.

Accuracy evaluation is to verify the accuracy of methods in the VU model. This test is exemplified by F_{edge_3} (green screen), F_{edge_5} (blurred screen), and F_{edge_8} (natural occlusion). We divided the experimental results into four kinds:

State1: A failure is detected as the failure one √.
State2: A failure is misdetected as the normal one ×.
State3: A normal one is detected as the normal one √.
State4: A normal one is misdetected as the failure one ×.

From Fig. 5(a), the methods in VU model have high accuracy rate, 0.77, 0.77, 0.97 in state1 and 0.92, 0.71, 0.80 in state3 (accurate evaluations) for F_{edge_3}, F_{edge_5}, and F_{edge_8}, respectively, which is much higher accuracy than those in state2 and state4 (inaccurate evaluations).

Performance evaluation (see Fig.5(b)) takes the case of F_{edge_8} (natural occlusion). It is one of collaborative detection methods in VU model based on edge computing and cloud computing. Results of baseline present the time (25s) consumed by the human operation. The method in our VU model achieves about 800% per cent improvement (2.778s).

Storage saving evaluation uses F_{edge_4} (black screen) as an example. There is a failure of black screen occurs during the normal video data stream. The existing approaches upload the video data with this failure (about 26 MB in Fig.5(c)) and store it in the cloud. However, the portion of video data with failure of black screen is removed from the video data by means of VU model's method, which employes edge computing in the camera. Storage saving for video data in the cloud can be achieved by around about 31 percent.

6 SUMMARY

In this paper, we firstly proposed the Video Usefulness model (named as VU model) for large-scale video surveillance systems by edge computing and cloud computing. The VU model can effectively use video data to find a failure in edge cameras, end-users, cloud, or network to manage large-scale video surveillance systems. By the VU model, we summary three failure domains with failures, which are used to evaluate video data uselessness. VU model is verified in the following two aspects. Firstly, the mean time to repair (i.e., MTTR) is improved by the real-time failure detection algorithms based on edge computing or cloud computing. Secondly, useless video data is directly handled by the edge rather than being uploaded to cloud; meanwhile storage saving is achieved.

ACKNOWLEDGMENT

This work is supported in part by the National Natural Science Foundation of China under Grants 61702004, Anhui University Funding for Doctoral Research (J01003214), the National Natural Science Foundation of China under Grant No.61572209, the Natural Science Foundation of Anhui Province (1708085QF160), and Key Technology R&D Program of Anhui Province (1704d0802193).

REFERENCES

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and others. 2010. A view of cloud computing. *Commun. ACM* 53, 4 (2010), 50–58.
[2] Ning Chen, Yu Chen, Sejun Song, Chin-Tser Huang, and Xinyue Ye. 2016. Smart Urban Surveillance Using Fog Computing. In *Edge Computing (SEC), IEEE/ACM Symposium on*. IEEE, IEEE Computer Society, Los Alamitos, CA, USA, 95–96.
[3] Ning Chen, Yu Chen, Yang You, Haibin Ling, Pengpeng Liang, and Roger Zimmermann. 2016. Dynamic urban surveillance video stream processing using fog computing. In *Multimedia Big Data (BigMM), 2016 IEEE Second International Conference on*. IEEE Computer Society, Los Alamitos, CA, USA, 105–112.
[4] Aakanksha Chowdhery, Paramvir Bahl, and Tan Zhang. 2017. BANDWIDTH EFFICIENT VIDEO SURVEILLANCE SYSTEM. (March 16 2017). US Patent 20,170,078,626.
[5] Thomas Tilden Collipi and David F Harvey. 2008. Method and apparatus for analyzing surveillance systems using a total surveillance time metric. (2008). US Patent 7,436,295.
[6] Gary Edmond and Mehera San Roque. 2013. Justicia's Gaze: Surveillance, Evidence and the Criminal Trial. *Surveillance & Society* 11, 3 (2013), 252–271.
[7] M Shamim Hossain. 2014. QoS-aware service composition for distributed video surveillance. *Multimedia tools and applications* 73, 1 (2014), 169–188.
[8] Tiejun Huang. 2014. Surveillance video: The biggest big data. (2014), 44–48 pages. http://www.computer.org/web/computingnow/archive/february2014
[9] Kinjal A Joshi and Darshak G Thakore. 2012. A Survey on Moving Object Detection and Tracking in Video Surveillance System. *International Journal of Soft Computing and Engineering* 2, 3 (2012), 44–48.
[10] Takuma Kon, Noriki Uchida, Koji Hashimoto, and Yoshitaka Shibata. 2012. Evaluation of a Seamless Surveillance Video Monitoring System Used by High-Speed Network and High-Resolution Omni-directional Cameras. In *Network-Based Information Systems (NBiS), 2012 15th International Conference on*. IEEE, IEEE Computer Society, Los Alamitos, CA, USA, 187–193.
[11] Abhaykumar Kumbhar, Farshad Koohifar, Ismail Guvenc, and Bruce Mueller. 2016. A survey on legacy and emerging technologies for public safety communications. *IEEE Communications Surveys & Tutorials* 19, 1 (2016), 97–124.
[12] Mingfu Li and Chen-Yu Lee. 2015. A cost-effective and real-time QoE evaluation method for multimedia streaming services. *Telecommunication Systems* 59, 3 (2015), 317–327.
[13] Shancang Li, Li Da Xu, and Shanshan Zhao. 2015. The internet of things: a survey. *Information Systems Frontiers* 17, 2 (2015), 243–259.
[14] Achmad Benny Mutiara. 2015. Internet of Things/Everythings (IoT/E). (2015).
[15] Hwin Dol Park, Ok-Gee Min, and Yong-Ju Lee. 2016. Scalable architecture for an automated surveillance system using edge computing. *The Journal of Supercomputing* 73, 3 (2016), 926–939.
[16] Tomi D Räty. 2010. Survey on contemporary remote surveillance systems for public safety. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 40, 5 (2010), 493–515.
[17] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge computing: Vision and challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646.
[18] S W Smith. 2004. Video surveillance system. (June 29 2004). US Patent 6,757,008.
[19] B Yogameena and K Sindhu Priya. 2015. Synoptic video based human crowd behavior analysis for forensic video surveillance. In *Advances in Pattern Recognition (ICAPR), 2015 Eighth International Conference on*. IEEE Computer Society, Kolkata, India, 1–6.
[20] Qingyang Zhang, Zhifeng Yu, Weisong Shi, and Hong Zhong. 2016. Demo Abstract: EVAPS: Edge Video Analysis for Public Safety. In *Edge Computing (SEC), IEEE/ACM Symposium on*. IEEE Computer Society, Los Alamitos, CA, USA, 121–122.