Video Object Detection Method Using Single-Frame Detection and Motion Vector Tracking

Masato Nohara
Graduate School of Science and Technology
Keio University
3-14-1 Hiyoshi, Kohoku, Yokohama, Kanagawa 223-8522, Japan
nohara@west.sd.keio.ac.jp

Hiroaki Nishi
Department of System Design, Faculty of Science and Technology
Keio University
3-14-1 Hiyoshi, Kohoku, Yokohama, Kanagawa 223-8522, Japan
west@sd.keio.ac.jp

***Abstract*—Video traffic on the Internet has been increasing rapidly and accounts for a large percentage of the total traffic. To process the increasing number of videos, edge computing is preferable for load balancing and bandwidth reduction. However, edge areas have less computational resources than cloud areas, and high-performance GPUs for processing videos at high speed are not always present. Therefore, a memory-saving and high-throughput video analysis method is necessary for analyzing videos in edge areas. In this paper, a video object detection method using single-frame detection and motion vector tracking is proposed. This method is classified as a pixel and compressed domain analysis method and is realized by compensating motion using the motion vectors that already exist in the compressed domain. This method is divided into two processes: CNN-based object detection and motion vector-based object detection. In addition, a network-transparent platform for video reconstruction in edge areas is constructed. The network-transparent service can be installed without modifying the existing end-device network settings, network configuration, and routing. The platform enables video object detection services to be added on without modification of these settings.**

Keywords—video object detection, moving detection, compressed domain analysis, motion vector, edge computing, network transparency

I. INTRODUCTION

Video traffic on the Internet has been increasing in recent years. A Cisco white paper [1] predicted that global Internet video traffic would reach 282.3 EB per month by 2022, up from 66.9 EB per month in 2017. This amounts to a compound annual growth rate of 33%. Video will account for 80% of all Internet traffic by 2022, up from 70% in 2017. In particular, Internet surveillance traffic is expected to increase 7-fold between 2017 and 2022. The increase in video traffic is due to several factors, such as improved image quality and increased penetration of services that involve video communication over the Internet.

An increasing number of videos are analyzed for object detection and tracking. Accordingly, the demand for video analysis is also increasing because analyzing surveillance camera videos automates manual tasks such as routine monitoring and pedestrians counting. The growing demand leads to an increase in the computational resources used to handle and analyze incoming video traffic from networks in a cloud computing node. To meet this demand, much computational and storage resources need to be allocated in the cloud environment to avoid overloading. However, it is difficult to analyze all the video traffic from the increasing number of end-devices in the cloud. In order to eliminate computational concentration in the cloud, the approach of processing videos in both the cloud and distributed edge areas was proposed.

It is hence crucial to process video in an edge computing environment for load balancing, bandwidth reduction, and network delay reduction. Additionally, video analysis at a faster frame rate than the video frame rate enables real-time video analysis. Since the frame rate of a general surveillance camera is about 30 FPS, a video analysis method with a speed of 30 FPS or more is desirable for real-time video analysis. However, edge areas have less computational resources than cloud areas, and high-performance GPUs that process images and videos at high speed are not always available in the edge areas because of their cost and power consumption. Thus, a method that can analyze videos in real-time without using high-performance GPUs is necessary. To address this problem, we use motion vectors in the compressed domain to track the object detected in the previous frame. Videos are generally stored and transmitted in compressed states using video codecs such as H.264 and H.265 that calculate motion vectors in order to compress the video data. The use of motion vectors reduces computational resources because it eliminates the need to recalculate motion information.

Moreover, it is desirable to perform network-transparent processing without modifying the end-device network settings, network configuration, routing, software design, and hardware configuration when processing videos in the edge areas. Additionally, network-transparent processing should be available on the network without affecting both end-devices and servers. Unless the edge nodes support network-transparent processing, it becomes necessary to change these settings, which increases the human labor costs.

In this paper, a video object detection method using single-frame detection and motion vector tracking is proposed. Video object detection is achieved in edge areas, where computational resources are scarce, and high-performance GPUs are not available. The proposed method aims to achieve real-time video analysis, even in an edge or IoT environment. Additionally, a network-transparent mechanism for acquiring videos in edge areas is constructed. Video object detection services can be added onto the network paths of operating video cameras that have already been installed without any modification to the network and camera settings. The contributions of this paper can be summarized in the following two points:

- The proposal of a memory-saving and high-throughput video object detection method without the use of high-performance GPUs.
- The construction of a network-transparent mechanism for acquiring video in the edge areas.

This paper is organized as follows. In Section II, related works are given. In section III, we explain our proposed method and its advantages in detail, and Section IV explains implementation of the proposed method. Section V compares the video object detection accuracy and throughput. Lastly, we summarize and discuss possible improvements and future work in the remaining sections.

II. Related Works

Due to the increase in demand for video analysis, various techniques, such as semantic segmentation and object tracking have emerged. Here, we classify video analysis methods based on the domains where the video is analyzed. Video analysis methods can be classified into the following three types:

- Pixel domain analysis
- Compressed domain analysis
- Pixel and compressed domain analysis

Widely used models, such as You Only Look Once (YOLO) [2][3], Single Shot Detector (SSD) [4], and Fast R-CNN [5] analyze videos in the pixel domain. These models are based on convolutional neural networks (CNNs), and are able to classify images and detect objects. These CNN-based models perform object detection for every frame.

Traditionally, region proposals are generated through selective search [6]. This takes a long time because many candidates are tried at the object position. The RoIPooling operation was proposed in Fast R-CNN for accelerating the regional feature extraction process. The Regional Proposal Network (RPN) was subsequently proposed in Faster R-CNN [7] to generate region proposals through the deep CNNs shared with Fast R-CNN. The one-stage object model directly predicts the interest-based bounding box based on the CNN-extracted feature map. Without an additional stage, a one-stage object detection model such as SSD and YOLO is usually faster than a two-stage object detection model.

Before explaining compressed domain analysis, we first describe video compression, which forms the basis of the proposed method. Videos are generally transmitted and stored in a compressed state and are typically compressed using video codecs such as H.264 and H.265 that use a technique called motion compensation. Motion compensation defines three types of frames, namely I-, P-, and B- frames, which hold different information. An I-frame is also called a key-frame, and it contains all the information of the image. In the key-frame, only intra-frame compression is performed. Unlike the I-frame, P-frames and B-frames contain motion vectors and, based on the I-frames, compensate for the movement so that the video can be played back. Both intra-frame and inter-frame compression are performed in these frames. A P-frame contains only the image difference information of the image, and during decoding, the video is restored from the difference information and the information in the I-frame. The I-frame is coded by performing backward prediction. A B-frame contains only the image difference information like the P-frame. During decoding, the video is restored from the difference information and the information of the I-frame encoded by either forward prediction, backward prediction, or bidirectional prediction.

Techniques for analyzing video in a compressed or partially expanded state, such as a method for detecting human faces [8], moving region segmentation [9], real-time object detection [10], and anomaly detection [11] have been developed. These techniques perform analysis using information such as motion vectors and macroblock types present in the compressed domain of the video. The information included in the compressed domain is calculated when the video is compressed. Thus, the recalculation of motion information is not required. Compared to analysis in the pixel domain, compressed domain analysis require less computational resources and can be performed with low memory requirements and high throughput.

A hybrid video object tracking method [12] and a motion vector-aided YOLO method (MV-YOLO) [13] that combine pixel and compressed domain analysis have been proposed. The hybrid video object tracking method uses motion vectors and macroblock type from the compressed domain based on a Markov Random Field (MRF) as well as color information from the pixel domain from a fully decoded I-frame. This hybrid framework provides better tracking accuracy than the state-of-the-art MRF model. MV-YOLO performs analysis in both the pixel and compressed domains to track a specific object. The object tracking accuracy is improved by using the results obtained by both motion vector processing and the YOLO object detection model. The use of both pixel and compressed domain information enables accurate object tracking. The object detection model is applied at every frame, and the object is tracked based on the detection result. Therefore, the throughput of MV-YOLO is lower than that of the object detection model itself.

Network transparency refers to the processing or installation of devices without modification of the existing end-device settings, configuration, and routing. In general, when the processing location in a network is changed, it is necessary to change the transmission destination of end-devices such as PCs and IoT devices, which incurs substantial human labor costs. On the other hand, it is not necessary to change the settings of the end devices when installing a network-transparent service to the network. For this reason, network-transparent services can be installed more easily than services that are not network-transparent. Additionally, network-transparent services can be processed on a network without affecting both end-devices and servers. The network transparent fog-based IoT platform [14] is a network-transparent service. This platform enables the addition of services to the network without modification of the end-device and network settings.

III. Proposed Method

We propose a video object detection method using single-frame detection and motion vector tracking. Here, the application environment of Fig. 1 is assumed. Videos are

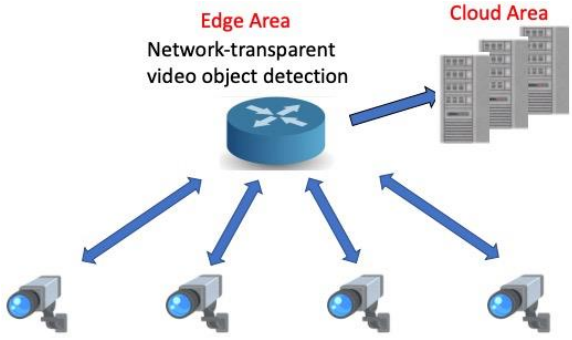

Fig. 1. Assumed environment

transparently acquired from an edge area with multiple surveillance cameras and analyzed. The cameras are connected to the Internet and send video data for storage in the cloud area continuously. For example, multiple surveillance cameras can be used to monitor roads around residential areas and production lines in a factory. The data obtained by the surveillance cameras passes through the edge area in the middle of the network. The application is installed in the edge node, acquires surveillance camera information transparently through the network, and detects objects. It is also assumed that computational resources are scarce in the edge area, and that it is difficult to mount high-performance GPUs. Thus, in the assumed environment, the use of a method that requires high-performance GPUs should be avoided. Therefore, we perform video object detection using the motion vectors already present in the compressed videos to realize memory-saving and high-throughput video object detection. Fig. 2 shows an overview of the proposed video object detection method. Object detection is performed through the processes shown in Fig. 2. There are about 30 frames per second in a video. These frames are subjected to two object detection processes, namely the CNN-based object detection and the motion vector-based object detection.

A. CNN-based Object Detection

CNN-based object detection is performed on a single frame using object detection models, such as SSD and YOLOv3. Any object detection model can be used for this process. The results of the object detection after the objects are detected are given as bounding boxes and labels. If an object that exists in the previous frame is not detected in the current frame, it is assumed that the object is in the same position as in the previous frame because objects do not actually disappear during the duration of a frame.

B. Motion Vector-based Object Detection

The detected objects in CNN-based object detection are tracked by using motion vectors from the compressed domain of the video. First, motion vectors in the compressed domain of the video are extracted. These motion vectors are vectors calculated to compress the video. Thus, these vectors do not entirely match the original motions of the objects in the video. In other words, motion vectors contain noise relative to the actual object motion. Therefore, we apply the median filter in Equation 1 to remove noisy vectors:

$$mv[x, y, t] = median\{mv[x', y', t'], x', y', t' \in w\} \quad (1)$$

In Equation 1, *mv* is the forward predicted motion vector, *w* the neighborhood pixels centered around location (x, y, t), x and y the spatial axes, and t the time axis. Since this step is computationally expensive, we used a filter with a size of $5 \times 5 \times 5$, as in Reference [11]. Backward predicted motion vectors in the B-frames are not used for tracking objects because backward prediction uses future frame information. The median filter sorts neighborhood vectors based on the vector lengths and determines the median value. Objects generally have continuous movements. This filter ensures that the motion vectors are spatiotemporally consistent and removes noisy vectors to preserve the movement continuity. Then, as shown in the lower row of Fig. 2, the position of the object is tracked based on the median-filtered motion vectors and the position of the object in the previous frame. Equation 2 shows how the position of the object is tracked. P_{now} and $P_{previous}$ are the positions of the object in the current and previous frames respectively, and $\overline{mv}$, which is given by Equation 3, is the averaged motion vector. In Equation 3, motion vectors are used to obtain the next position of the object. After estimating the position of the object, the result of the position estimation is given as a bounding box and label.

$$P_{now} = P_{previous} + \overline{mv} \quad (2)$$

$$\overline{mv} = average\{mv[w, h], w, h \in P_{previous}\} \quad (3)$$

Motion vector-based object detection cannot be performed on I-frames, since I-frames do not have motion vectors. Only CNN-based object detection can be performed for I-frames. In contrast, P-frames and B-frames have motion vectors in the compressed domain, and both CNN-based and motion vector-based object detection can be performed. In Fig. 3(a), CNN-based object detection is performed on two out of thirteen frames, and motion vector-based object detection on the other eleven frames. In Fig. 3(b), CNN-based object detection is performed on seven out of thirteen frames, and motion vector-based object detection is performed on the other six frames. The ratio of CNN-based to motion vector-based object detection can be changed freely. Therefore, it is possible to change the frequency of CNN-based object detection, which has a longer execution time and uses a larger amount of memory than motion vector-based object detection. Memory-saving and high-throughput object detection can be realized by reducing the frequency of CNN-based object detection. Additionally, there are trade-offs between throughput and accuracy. These trade-offs can be controlled by changing the balance of CNN-based and motion vector-based object detection.

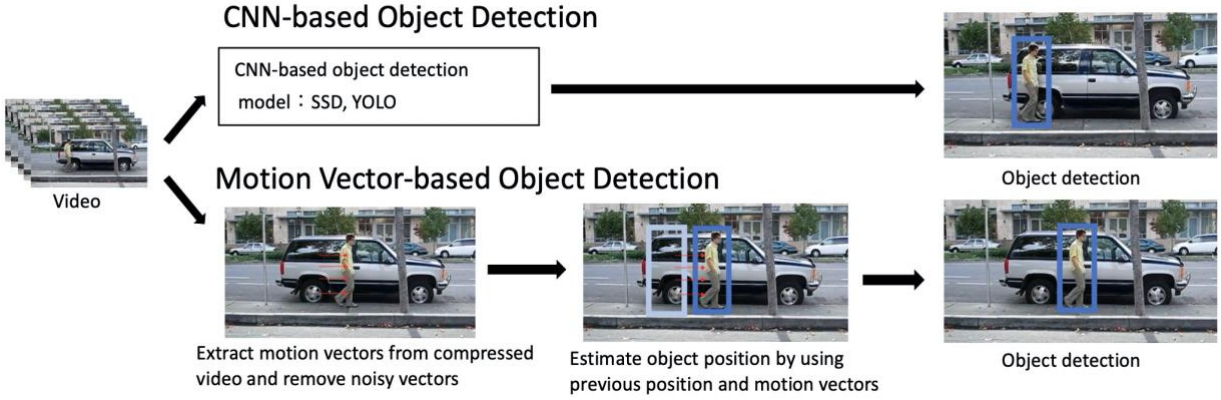

Fig. 2. An overview of the proposed video object detection method

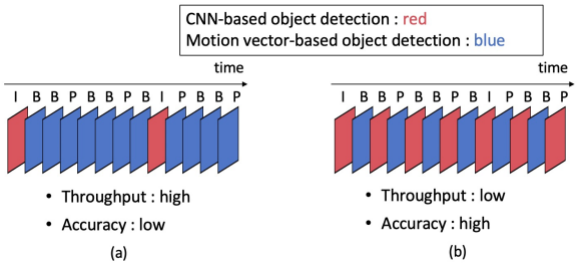

Fig. 3. Trade-offs between throughput and accuracy

C. Features

We summarize the key features of the proposed video object detection method.

High-throughput and memory-saving: CNN-based object detection is not performed for every frame, and motion information does not need to be recalculated for tracking objects because motion vectors already present in the video compressed domain. These result in sufficient throughput for real-time analysis, even in an environment without high-performance GPUs. Also, CNN-based object detection uses a larger amount of memory than motion vector-based object detection. When we take the time average of memory usage, the proposed method uses less memory than performing CNN-based object detection for all frames.

Trade-offs between throughput and accuracy: The trade-offs between accuracy and throughput can be controlled by changing the balance of CNN-based and motion vector-based object detection. Generally, a video object detection method can only give a certain accuracy and throughput. Some detection applications may require high accuracy, while other detection applications may require high throughput. Our proposed method can respond to various requirements in the trade-offs between throughput and accuracy.

Compatibility with models for object detection: Our object detection method is not dependent on any particular object detection model. We used YOLOv3 and SSD for the experiment. Besides these models, it is possible to use other object detection models such as FAST R-CNN and other versions of YOLO. If the accuracy of the object detection model increases, the accuracy of the proposed video object detection method will also increase.

IV. Implementation

In our implementation, the FFmpeg [15] and Data Plane Development Kit (DPDK) [16] libraries were used. FFmpeg is a complete cross-platform solution for recording, converting, and streaming audio and video. DPDK is a set of libraries and drivers for fast packet processing using kernel bypass technology. The structure of the application shown in Fig. 4 is described below.

A. DPDK Application

In the DPDK application, packets flowing through the network are stored in the Rx queue. The packets stored in the Rx Queue are sent to the Tx queue after processing by the DPDK application and transmitted again over the network. At the same time, the TCP stream is reconstructed in the DPDK application. When a video is transmitted using a video streaming protocol, it is divided into multiple files. Thus, the video is divided into multiple TCP streams. These TCP streams need to be grouped by video. The reconstructed TCP streams are grouped based on the IP address and port number.

B. FFmpeg Application

The video from the DPDK application is passed to the FFmpeg application. The video passed to the FFmpeg were used for the experiment. The video information includes moving image information and audio information. Since only the moving image information is required for video analysis, the audio information is removed. The video is then decoded using a suitable decoder from libavcodec, which is a library in FFmpeg. The resultant video frames from the processed video can then be analyzed.

In the experiment, a Shuttle DH310 and an Intel NUC, as shown in Fig. 5, were used. Their specifications are shown in TABLE I. The dimensions of the Shuttle DH310 are 190 × 165 × 43 mm, and those of the Intel NUC are 102 × 102 × 28 mm. The sizes of these devices are smaller than general computing devices. The edge areas are scattered across many places compared to the cloud areas, and have limited computational resources and available space. To model such environments, small devices were used for the experiment.

These devices were arranged as shown in Fig. 5. The Intel NUC was used as a device for delivering videos from sources such as a surveillance camera. The Shuttle DH310 was used as a device for processing videos in a network-transparent manner in the edge area and for storage in the cloud area. The arrow, shown in Fig. 5, indicates the direction in which the video information was transmitted.

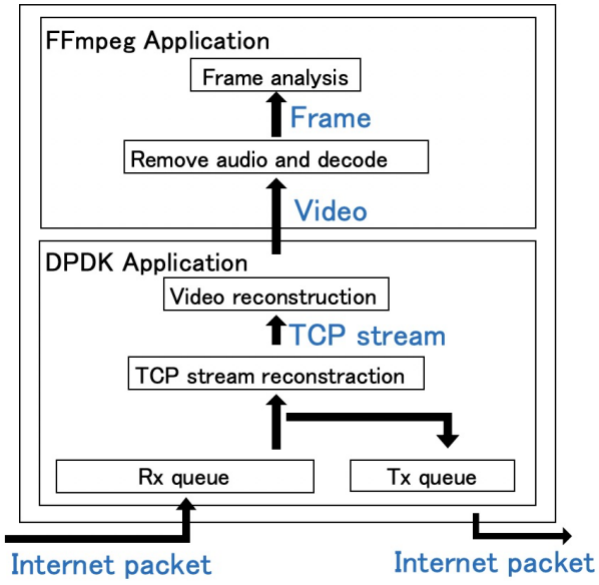

Fig. 4. Application implementation

Fig. 5. Device placement and video transmission direction

TABLE I. Simulation Environment

Device	Intel NUC	Shuttle DH310
OS	Ubuntu 18.04.1 LTS	Ubuntu 18.04.1 LTS
CPU	Intel(R) Core(TM) i3-6100U CPU@2.30GHz	Intel(R) Core(TM) i7-8700CPU@3.20GHz
Memory	8 G	32 G
Size	102 x 102 x 28 mm	190 x 165 x 43 mm

V. Evaluation and Results

First, we confirmed that the filtered motion vectors match the original motion of objects by using videos from surveillance cameras. Fig. 6 is a set of images from surveillance cameras after applying a median filter to the motion vectors shown with coloring. 75% accuracy was maintained when compared against moving object detection by taking image differences in the video.

TABLE II shows the moving object detection throughput. The moving object detection throughput from frame differences was 45.4 FPS, and that from motion vectors was 132.4 FPS. When the motion was detected using the motion vectors, the throughput was 2.92 times higher than when the motion was detected by taking the frame differences.

Fig. 7 shows the memory usage during moving object detection. Moving object detection using motion vectors, indicated by the black line, used about 7 Mbytes of memory on average. In comparison, moving object detection by taking the differences between frames, indicated by the blue line, used about 200 Mbytes of memory on average. By using motion vectors, the memory usage was reduced. When taking the differences between frames, multiple frames were expanded in the memory, and the differences were calculated based on information from multiple frames. In contrast, when motion vectors were used, there was no need to expand multiple frames in the memory because the motion information already existed in each frame. For these reasons, moving object detection method using motion vectors requires less memory than moving object detection method using the differences between frames. The throughput of moving object detection was 2.92 times higher, as shown in the throughput results, and the duration of memory usage was shorter for moving object detection from the motion vectors compared to moving object detection from the differences between frames. This means that it is possible to analyze videos from more surveillance cameras at once.

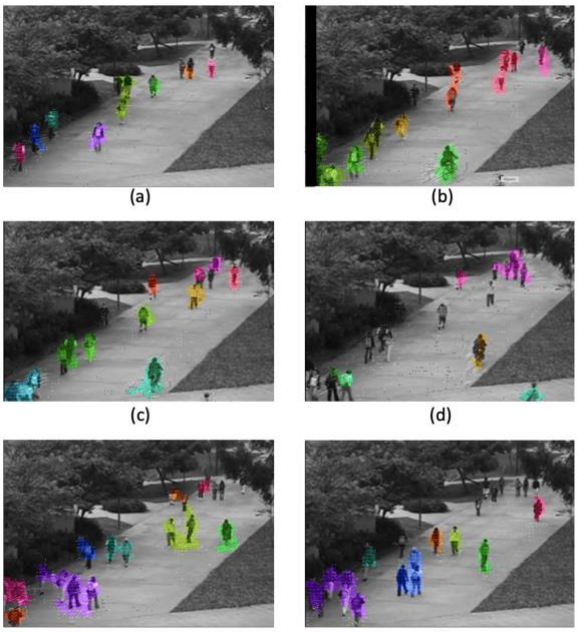

Fig. 6. Moving object detection

TABLE II. Throughput of Moving Object Detection (FPS)

Moving object detection by frame differences (a)	45.4
Moving object detection by motion vectors (b)	132.4
Throughput ratio (a/b)	2.92

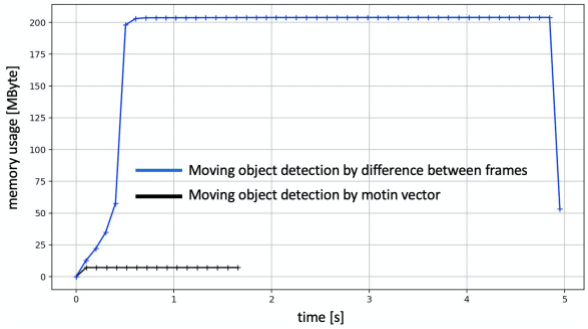

Fig. 7. Memory usage of moving object detection

Fig. 8 shows part of the video object detection results. In the video, there are multiple cars and walking humans. The results of the car and human detection are depicted with bounding boxes and labels.

$$IoU = \frac{P_{gt} \cap P_{now}}{P_{gt} \cup P_{now}} \quad (4)$$

$$mAP50 = \frac{number\ of\ objects\ whose\ IoU \geq 0.5}{number\ of\ objects} \quad (5)$$

The detection accuracy was evaluated using *mAP50*, the ratio of the object for which the Intersection over Union (IoU) is bigger than 0.5 to the number of objects, as given in Equation 5. The *IoU* is given in Equation 4. In Equation 4, P_{gt} is the position of an object, and P_{now} is the estimated position of the object. If the estimated label of the object is different from the actual label of the object, the *IoU* is determined to be 0.

As TABLE IV shows, the *mAP50* was 0.796 when the detection was performed using the SSD object detection model on all the frames, whereas it was 0.703 when the detection was performed using SSD only on the I-frames. The *mAP50* was 0.773 when the objects were detected using YOLOv3 for all frames, whereas it was 0.704 when the objects were detected using only the I-frames. Compared to detection using the object detection model on all frames, detection using only the I-frames has a lower average *IoU* per frame. However, all the *mAP50* values were at least 0.7, so a certain accuracy level was still maintained.

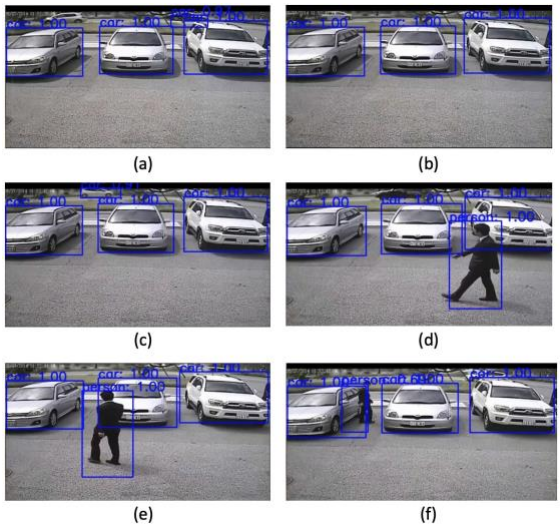

Fig. 8. Video object detection

TABLE III. THROUGHPUT OF VIDEO OBJECT DETECTION (FPS)

	SSD	YOLOv3
CNN-based object detection for all frames (a)	1.3	4.7
Proposed method (b)	58.3	75.5
Throughput ratio (a/b)	44.5	16.1

TABLE IV. mAP50 OF VIDEO OBJECT DETECTION

	SSD	YOLOv3
CNN-based object detection for all frames (a)	0.796	0.774
Proposed method (b)	0.703	0.705
Accuracy ratio (a/b)	0.88	0.91

TABLE III shows the video object detection throughput results. When SSD was applied to all the frames, the throughout was 1.31 FPS, whereas it was 58.3 FPS when SSD was applied to only the I-frames. When YOLOv3 was used, the throughputs were 4.69 FPS and 75.52 FPS, respectively. Regardless of which object detection model was used, object detection using only I-frames and using motion vectors for the other frames was more efficient than using the object detection model for all the frames.

Fig. 9 shows the trade-offs between video object detection accuracy and throughput. The smaller the number of detections using the object detection model, the higher the throughput. The accuracy is shown through the *mAP50*. The smaller the number of detections using the object detection model, the lower the accuracy. On the other hand, the accuracy was only slightly degraded even when the number of detections using the object detection model was reduced from 300 to about 50 in a video composed of 300 frames. Accordingly, the object detection accuracy can be maintained by estimating the position of the object in the next frame using motion vectors based on the object detection result. Moreover, the accuracy and throughput can be controlled by changing the frequency of CNN-based and motion vector-based object detection.

Fig. 10 shows the memory usage of the video object detection. The maximum memory usages of the two methods were almost the same because both methods use a CNN-based object detection model. Whereas, when the time average was taken, the average memory usage using the object detection model for all frames was about 500 Mbyte and that of detection using the object detection model for only I-frames was about 390 Mbyte. Consequently, using the object detection model on only I-frames used about 100 Mbyte less memory on the average. Detecting objects using the CNN-based object detection model require the execution of processes such as convolution and pooling. These processes

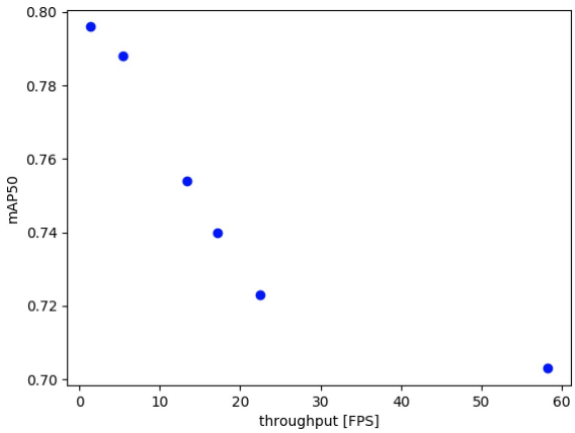

Fig. 9. Trade-offs between throughput and accuracy

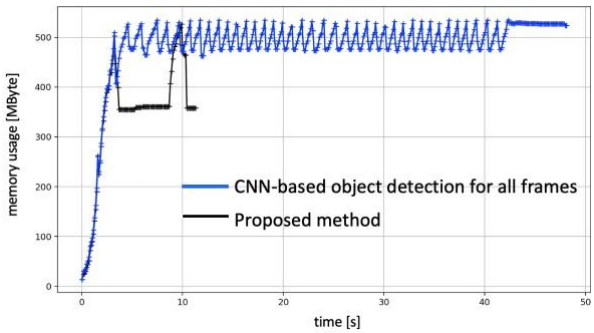

Fig. 10. Memory usage during video object detection

use more memory than the processes involved when motion vectors are used. When motion vectors are used, the video object detection throughput is several times higher, as shown in the throughput results, and the duration of memory usage becomes shorter than CNN-based object detection for all frames. This means that it is possible to detect objects from more surveillance cameras at once.

VI. CONCLUSION

In this paper, we proposed a video object detection method using single-frame detection and motion vector tracking. The contributions of this paper described in section I are reiterated below:

- A video object detection method was proposed for memory-saving and high-throughput analysis without the use of high-performance computing nodes like GPUs.
- A network-transparent mechanism for acquiring video was constructed in edge areas.

The proposed video object detection method uses motion vectors contained in the compressed domain of the videos. The results in Section V show that the throughput of the method using motion vectors is 2.92 times and the memory usage 0.04 times those of using the frame differences for motion detection. For video object detection, the throughput was 44.5 times, and the memory usage was 0.78 times those of analyzing videos at the pixel domain using CNN-based object detection models. In addition, the videos were obtained in a network-transparent manner at the edge areas. The video object detection service can be added onto the network paths of cameras that are already installed and working without any modification to the network and surveillance camera settings. Therefore, by using the motion vectors included in the video compressed domain, it is possible to perform memory-saving and high-throughput video object detection in edge areas. As a result, it has become possible to not only process the increasing amount of videos intensively in the cloud, but also to process videos in the edge areas and exploit the advantages of edge computing in video analysis.

VII. FUTURE WORK

In future work, the accuracy will be improved by using more sophisticated motion vector analysis, and the maximum number of objects that can be recognized will be increased. Variable camera angles will be supported by considering the movement of both the object and the background as the camera moves.

ACKNOWLEDGMENT

This work was supported by JST CREST Grant Number JPMJCR19K1, and the commissioned research by National Institute of Information and Communications Technology (NICT, Grant Number 22004) , JAPAN.

REFERENCES

[1] "Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper - Cisco." [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html. [Accessed: 08-Feb-2020].

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[3] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018.

[4] W. Liu *et al.*, "SSD: Single shot multibox detector," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.

[5] R. Girshick, "Fast R-CNN," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.

[6] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, 2013, doi: 10.1007/s11263-013-0620-5.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[8] S. Ranjbar Alvar, H. Choi, and I. V. Bajić, "Can You Tell a Face from a HEVC Bitstream?," *Proc. - IEEE 1st Conf. Multimed. Inf. Process. Retrieval, MIPR 2018*, vol. 1, pp. 257–261, 2018, doi: 10.1109/MIPR.2018.00060.

[9] K. S. Devi, N. Malmurugan, and H. Ambika, "Moving region segmentation from compressed video using global motion estimation by macroblock classification and markov random field model," *2013 IEEE Int. Conf. Emerg. Trends Comput. Commun. Nanotechnology, ICE-CCN 2013*, no. Iceccn, pp. 163–167, 2013, doi: 10.1109/ICE-CCN.2013.6528484.

[10] R. C. Moura, E. M. Hemerly, and A. M. Cunha, "Temporal Motion Vector Filter for Fast Object Detection on Compressed Video," *J. Commun. Inf. Syst.*, vol. 29, no. 1, pp. 12–24, 2014, doi: 10.14209/jcis.2014.1.

[11] S. Biswas and R. V. Babu, "Anomaly detection in compressed H.264/AVC video," *Multimed. Tools Appl.*, vol. 74, no. 24, pp. 11099–11115, 2015, doi: 10.1007/s11042-014-2219-4.

[12] S. Gul, J. T. Meyer, C. Hellge, T. Schierl, and W. Samek, "Hybrid video object tracking in H.265/HEVC video streams," *2016 IEEE 18th Int. Work. Multimed. Signal Process. MMSP 2016*, 2017, doi: 10.1109/MMSP.2016.7813363.

[13] S. R. Alvar and I. V. Bajic, "MV-YOLO: Motion vector-aided tracking by semantic object detection," *2018 IEEE 20th Int. Work. Multimed. Signal Process. MMSP 2018*, 2018, doi: 10.1109/MMSP.2018.8547125.

[14] R. Morishima and H. Nishi, "Network Transparent Fog-based IoT Platform for Industrial IoT," pp. 920–925, 2020, doi: 10.1109/indin41052.2019.8972178.

[15] "FFmpeg." [Online]. Available: http://ffmpeg.org/. [Accessed: 13-Feb-2020].

[16] "DPDK." [Online]. Available: https://core.dpdk.org/. [Accessed: 13-Feb-2020].