J Inf Process Syst, Vol.17, No.2, pp.227~241, April 2021
https://doi.org/10.3745/JIPS.02.0154

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

# RAVIP: Real-Time AI Vision Platform for Heterogeneous Multi-Channel Video Stream

Jeonghun Lee* and Kwang-il Hwang*

**Abstract**

Object detection techniques based on deep learning such as YOLO have high detection performance and precision in a single channel video stream. In order to expand to multiple channel object detection in real-time, however, high-performance hardware is required. In this paper, we propose a novel back-end server framework, a real-time AI vision platform (RAVIP), which can extend the object detection function from single channel to simultaneous multi-channels, which can work well even in low-end server hardware. RAVIP assembles appropriate component modules from the RODEM (real-time object detection module) Base to create per-channel instances for each channel, enabling efficient parallelization of object detection instances on limited hardware resources through continuous monitoring with respect to resource utilization. Through practical experiments, RAVIP shows that it is possible to optimize CPU, GPU, and memory utilization while performing object detection service in a multi-channel situation. In addition, it has been proven that RAVIP can provide object detection services with 25 FPS for all 16 channels at the same time.

**Keywords**

Multi-Channel, Multi-Stream, Object Detection, Surveillance Systems, Vision

## 1. Introduction

Beginning with CCTV based on black-and-white video in the 80s, through the era of magnetic tape storage of low-quality video in the 90s, and in the 2000s, the prevalence of digital video recorder (DVR) [1], which converts analog CCTV video to digital and stores it on disk, was activated. Since the mid-2000s, network video recorder (NVR) [2] has been mainly used for data storage and monitoring along with the spread of IP cameras. These DVRs and NVRs are technologies that enable monitoring and storage/playback on a single screen by receiving multiple CCTV inputs, and have recently been used in almost all CCTV usage sites. Recently, due to the rapid development of image processing technology, by combining various services beyond basic functions, DVR and NVR are rapidly evolving into real-time intelligent monitoring systems. The most basic technology in such an intelligent monitoring service is to detect various objects including people in a screen (frame) from an image. Traditional object detection algorithms—HOG [3], SIFT [4], and HARR [5]—have been used for a long time, but recently object detection technologies using deep learning technology based on convolutional neural networks—R-CNN [6], YOLO Family [7-9], and Faster R-CNN [10]—showed significantly superior performance


Manuscript received July 29, 2020; first revision January 5, 2021; second revision February 15, 2021; accepted March 5, 2021.
**Corresponding Author:** Kwang-il Hwang (hkwangil@inu.ac.kr)
* Dept. of Embedded Systems Engineering, Incheon National University, Incheon, Korea (junghoon450@gmail.com, hkwangil@inu.ac.kr)

and generalization compared to traditional object detection technology.

Thus, the development of various applications and services based on this object detection technology is accelerating. In particular, YOLO (you only look once) is used as the main software for object detection in various applications by showing more precise and faster performance than other techniques through continuous version maintenance. Nevertheless, a system with a powerful GPU is required to properly perform the performance of YOLO. It is impossible, however, to directly apply deep learning-based object detection software such as YOLO to existing DVRs and NVR systems because most of the currently deployed DVRs and NVRs do not have such a powerful GPU.

Therefore, in order to use this real-time object detection service in DVR or NVR, YOLO must be running on a separate back end server equipped with a GPU. But in the case of NVR and DVR, since multiple-channel streams must be processed in real-time, real-time object detection from multi-channel streams must be possible in the back-end server. Basically, since YOLO was developed for object detection for single channel video input on a single GPU, it is necessary to mount multiple GPUs on the back end server in order to expand to this multi-channel. Still, this is not an appropriate solution because it causes problems that dramatically increase the cost of the server.

Therefore, this paper proposes a real-time AI vision platform (RAVIP), a real-time multi-channel object detection service framework in a back-end system with a single GPU capable of association with a DVR/NVR that stores multi-channel video streams. The proposed method not only enables real-time multi-channel object detection service in a single system by parallelizing per-channel YOLO instances at a high level, but also allows multiple instances to be executed in one system, by maximizing utilization of resources such as CPU, GPU, and memory.

The organization of this paper is as follows. Section 2 analyzes research on object detection for multi-channel images. Section 3 provides a brief overview of the RAVIP proposed in this paper, and Section 4 describes in detail the structure, features, and service flow of RAVIP. Section 5 presents the actual experimental results and performance evaluation of RAVIP, and this paper concludes in Section 6.

## 2. Related Work

As the importance of processing a plurality of video streams has increased, research on the subject matter has been actively conducted recently. First, Parsola et al. [11] proposed a Hadoop system architecture for storing large-scale multi-stream video from multiple cameras and finding useful information therefrom. Also Karimi-Mansoub et al. [12] proposed a technique for detecting objects by YOLOv3 from multiple streams on a single GPU through the multi-thread structure using MUX and DeMUX. Tan et al. [13] proposed a technique that tracks a specific object at an actual location by using yolo in parallel to track a specific object from multiple streams and predicting location information of a specific target using LSTM (long short-term memory). Kim and Park [14] proposed an efficient cooperation framework between the edge camera and the video analysis server. Mattela et al. [15] proposed a new EcDNN (Enterprise class deep neural network) that can simultaneously perform object detection and face recognition in the surveillance system, based on a DNN structure constructing through a multi-task learning architecture.

Park et al. [16] implemented a video analysis method using a GPU using a stochastic data association filter for multi-object tracking, and proposed a technique of applying ResNet for object detection and a

stochastic data association filter for object tracking. Wu et al. [17] proposed a system for analyzing soccer matches using multi-cameras, and Aslan and Ileri [18] performed performance evaluation on a mobile processor for multi-thread-based facial recognition technology. Wang [19] proposed an efficient structure for the object detection pipeline in the GPU using CUDA, and Liu et al. [20] proposed a parallel object tracking technique for illumination variation. In addition, the DeepStream [21] based on Gstreamer provides SDKs that can implement multi-channel object detection/tracking technology based on Linux and various embedded AI processors in the form of plug-ins. Table 1 summarizes the characteristics of the related studies mentioned so far and their differences from this study.

**Table 1.** Summary of related work

| Study | Characteristics | Differentiation or Limitation |
|---|---|---|
| Parsola et al. [11] | Hadoop system architecture for storing large-scale multi-stream video from multiple cameras and finding useful information | Lack of an object detection service from real-time video |
| Karimi-Mansoub et al. [12] | Multi-channel video processing based on a single GPU with multi-thread | Test results only on four streams |
| Tan et al. [13] | Parallel YOLO processing and specific target tracking using LSTM | Lack of object detection from multiple streams |
| Kim and Park [14] | Collaboration framework between the edge camera and the video analysis server | Lack of details about efficient processing of multiple streams |
| Mattela et al. [15] | EcDNN (Enterprise Class DNN) that can simultaneously perform object detection and face recognition in the surveillance system | Focus only on simultaneous processing of face detection and object detection.<br>Lack of object detection from multiple streams |
| Park et al. [16] | Video analysis method using a stochastic data association filter for multi-object tracking | Focus on multiple object tracking<br>Lack of object detection from multiple streams |
| Wu et al. [17] | System for analyzing soccer matches using multi-cameras | Focus on the specific application, soccer<br>Lack of generalization |
| Wang [19] | Object detection pipeline in the GPU using CUDA | Difficulties in implementation in terms of application design<br>Lack of multiple camera interface methods |
| Liu et al. [20] | Parallel object tracking technique for illumination variation | Focus on multiple object tracking<br>Lack of object detection from multiple streams |
| DeepStream [21] | Multi-channel object detection and tracking technology based on Linux and various embedded AI processors in the form of plug-ins | Deterioration of overall object detection performance due to the increase of sources through multiplexing processing of multi-channel inputs |

## 3. Overview of RAVIP Service

In order to enable real-time object detection service from multi-channel video streams, we propose real-time AI vision platform (RAVIP). RAVIP provides a flexible local back end service that can be associated with existing video storage systems such as DVR and NVR. In addition, it can be used in various applications by supporting various types of cameras. The internal software is designed with a completely modular structure, and as shown in Fig. 1, it is easy to create a new RODEM instance by

assembling the appropriate components of the real-time object detection module (RODEM) base. RODEM relies on the performance of the GPU installed in the server, but since it is basically designed independently of the hardware, it can be executed only by installing software on various systems without specialized hardware-related configuration.

Basically, multiple channels of object detection are possible at the same time, and in the RTX2080 TI GPU environment, object detection for 16 channels of simultaneous stream is supported at 15 to 30 FPS. In addition, by supporting various standard streaming protocols such as Real-time Transport Protocol (RTP) [22] and Real-Time Streaming Protocol (RTSP) [23], various video sources can be accepted as input.

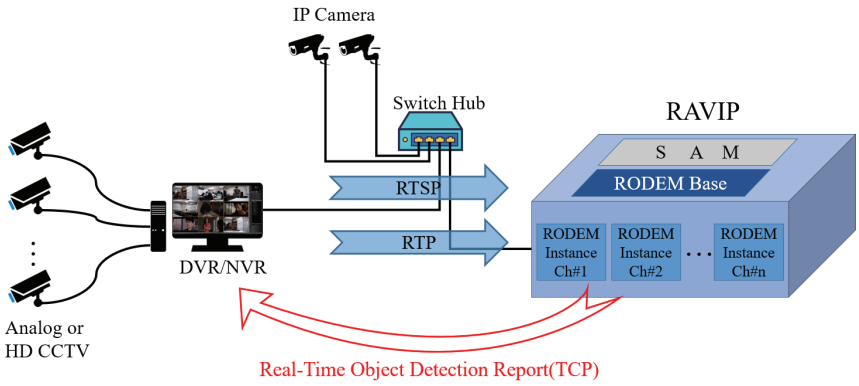



**Fig. 1.** Overall system architecture of RAVIP.

## 4. Design of Real-Time AI Vision Platform

This section introduces the detailed structure of the proposed RAVIP and presents how RAVIP efficiently detects objects from multiple streams in real time. As shown in Fig. 1, RAVIP is a back-end local sever that associated with NVR and DVR. There are two main stream input sources to RAVIP. First, in case of IP camera input, the same video source as DVR and NVR is received through RTSP through the switch hub. Second, in case of analog CCTV, the video received from the dedicated DVR is input to RAVIP through RTP. Each RODEM instance inside RAVIP is executed for each of these input source channels to perform object detection for that channel, and transmits metadata for the object detection in real time through a separate control channel.

Thus, it is possible to receive object detection service while performing the existing CCTV monitoring and storage functions in DVR and NVR. RAVIP is largely composed of SAM (server access manager) and RODEM Base, and per-channel RODEM instance which is created for each service request of the client channel through SAM. The per-channel RODEM instance enables multiple channel object detection services at the same time through parallel processing.

### 4.1 Server Access Manager

RAVIP basically starts service by requesting object detection for the channel from DVR and NVR. SAM plays a role of creating and managing a dedicated object detection instance for a new channel by

receiving a request from a client (DVR or NVR) to the RAVIP server. As shown in Fig. 2, RAVIP basically waits as a server. When a service request message is received from a client, it first performs authentication for the client and creates a new instance based on RODEM Base when authentication is completed. The authentication used in RAVIP is a process of checking whether the client is a user already registered in the server, and authentication is performed simply by checking the user ID and password.

In the created instance, the image of the corresponding channel is received in real time, and object detection is performed, and metadata for each object is transmitted to the client in real time.

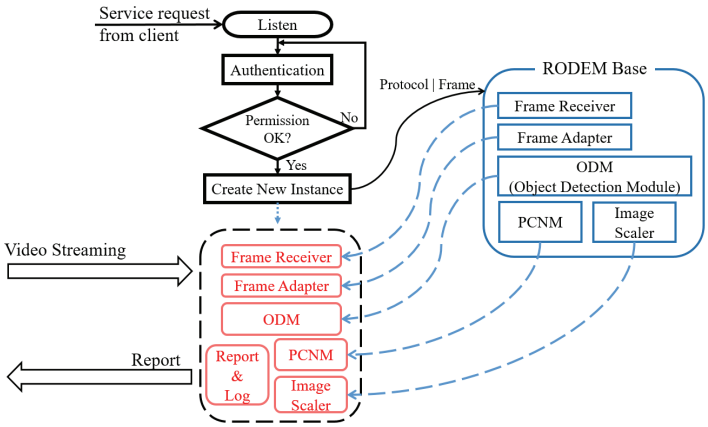



**Fig. 2.** Server access manager.

## 4.2 RODEM: Real-Time Object Detection Module Base

Fig. 3 shows the structure of RODEM Base. The new instance created by the client's request performs the corresponding service by assembling each module based on the RODEM Base. RODEM Base has a Frame Receiver module for receiving the video of the corresponding channel. In this module, all frames using RTP or RTSP can be received. The frame received through each frame receiver controls the rate of the frame delivered to the AI module through the Frame Adapter module. If a frame is received at a rate higher than the processing time in the AI module, it includes a function to drop the frame queue so that the AI module can process it. The AI module is a core service of RAVIP, and it can selectively use services such as object detection and face recognition.

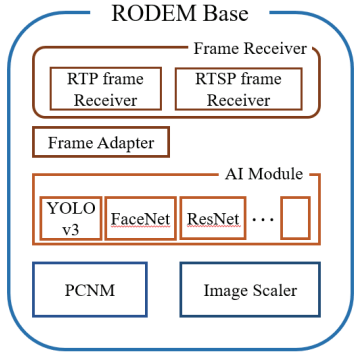



**Fig. 3.** RODEM Base.

YOLOv3 and ResNet16 can be basically selected as object detection service, and FaceNet can be used for face recognition. Nonetheless, in this paper, it is based on human object detection customized training with YOLOv3. Per-channel network manager (PCNM) is responsible for transmitting the object detection results in real time to the client in the corresponding channel. When SAM creates a new RODEM instance for a new channel, a new control channel for it is created, and this channel is managed by PCNM. In addition, the size of the bounding box must be rescaled again according to the size of the image of the original frame because each object detection module accepts an image of a certain size and returns the object detection result. To this end, the image scaler module converts the bounding box to a size that fits the original frame.

## 4.3 Per-Channel RODEM Instance

Fig. 4 shows the structure of how a new object detection instance is created and executed by the client's service request, and how the object detection result can be reported in real time for each channel. First, upon the service request of the client, SAM creates a new per-channel instance from the RODEM Base. At this time, SAM checks whether the corresponding video transmission protocol is RTP or RTSP in the service request and executes the frame receiver suitable for this in the RODEM Base. The new instance receives the video frame in real time through the address information of the RTP or RTSP. The received frames are sent to the AI module to process each frame through the Frame Adapter. At this moment, the output rate of the received frame queue is adjusted based on feedback on the execution time of the AI module so that delays caused by AI module do not accumulate. Object information recognized in each frame through AI module (YOLO) is transmitted to the client as bounding box information.

When creating a per-channel instance, the link to this is already connected to the client, and the real-time object detection result is converted into an optimized result through the image scaler and transmitted to the client through PCNM. By using the result of the bounding box in the video for the same channel (drawing and storing metadata), the NVR and DVR can perform the object detection service by the back-end service naturally.

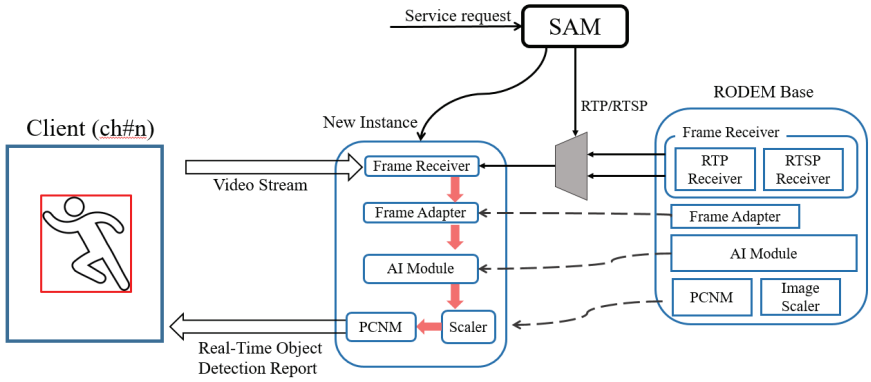



**Fig. 4.** How to create new RODEM instance.

## 4.4 Application Level Parallel Processing for Multiple RODEM Instances

RAVIP basically runs on a server equipped with one or more GPUs. In order to use such a GPU more flexibly, RAVIP uses GPU resources distributed at the application level rather than at the lower level

GPU allocation. Fig. 5 shows an example of application level parallel processing of multiple RODEM instances. RODEM instances created by multiple client requests are executed as independent threads and provide object detection services for video channels of each client. By monitoring the currently serviced FPS in real time through the Frame Adapter module in each RODEM instance and controlling the serviced frame rate, GPU resources used simultaneously in multiple instances are efficiently distributed. Since this application-level resource utilization technique operates independently on the hardware of the server, the performance of RAVIP can be further improved by the expansion of the server hardware without further configurations.

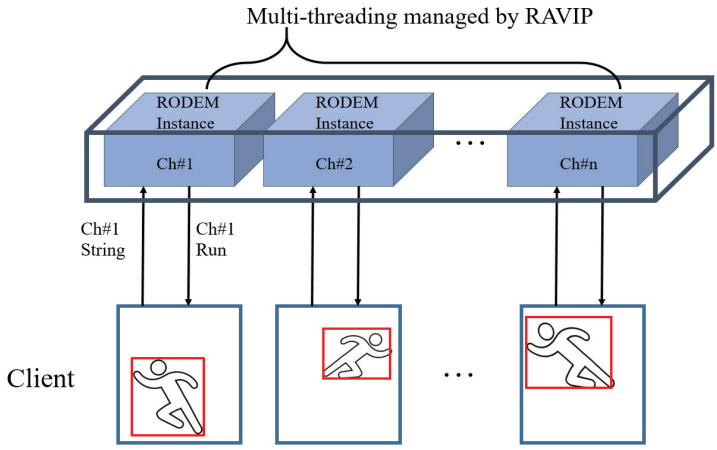



**Fig. 5.** Parallel RODEM processing.

## 4.5 RAVIP Service Flow

Fig. 6 shows the service flow between the client (DVR, or NVR) and the RAVIP server. First, the client sends a service request for the channel to receive the service to RAVIP. In contrast, SAM performs authentication and creates a new instance from RODEM Base based on the service type (RTP or RTSP). Information about the new instance (NEW SOCKET) is responded to the client. The client establishes a back-channel connection with the corresponding instance by requesting an instance connection request to receive the service of the corresponding channel. The instance of the channel receives real-time video streaming, analyzes it in real time in the AI module, and reports the result (object type and bounding box info) in real time through the back channel. The client displays and saves the object detection contents based on the result received from RAVIP. These services can request and terminate services for new channels at any time through SAM.

As shown in Fig. 6, the client first transmits a service request for channel 1 to the RAVIP server. After simple authentication, the server sends a response message for New_Instance_info to the client. Based on this information, the client sends an instance_connection_request for channel 1 to the server again, and the corresponding RODEM instance of the server sends a response to establish a service connection for channel 1. Real-time video streaming transmitted from then on is processed by the ROEOM server, and a report on the object detected in real time is transmitted to the client. Likewise, a client can receive simultaneous object recognition services for multiple channels by transmitting service_request for multiple channels.

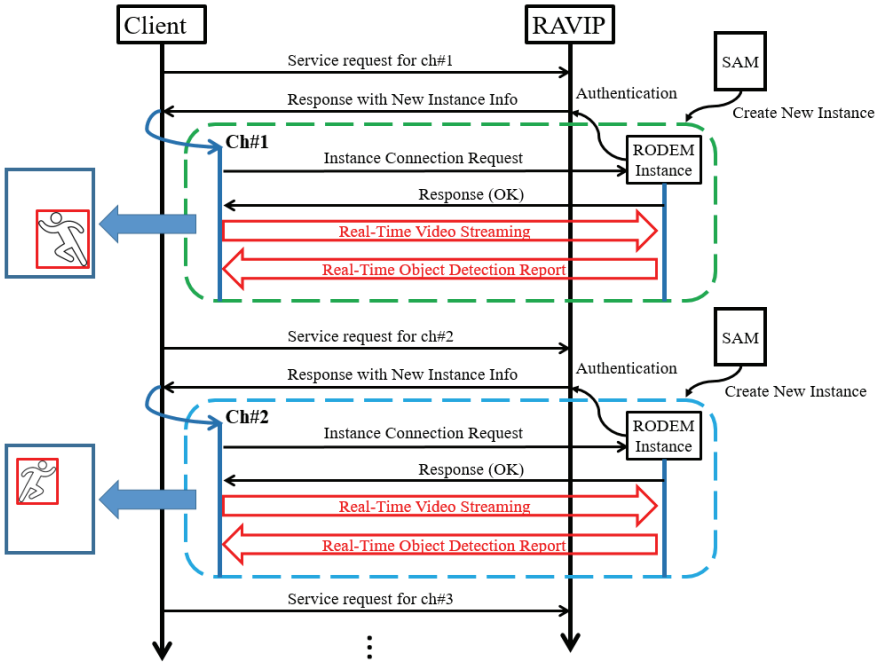



**Fig. 6.** RAVIP service flow.

# 5. Experimental Results

To test RAVIP's ability to detect multi-channel object in real time, we developed back-end server software that is associated with real NVRs. The developed RAVIP runs in a Windows environment, and its performance is evaluated on hardware with two different specifications. Frame rate was measured for YOLO object detection from multiple channel video sources, and the usage of CPU, GPU, and memory resources in multiple channels was observed.

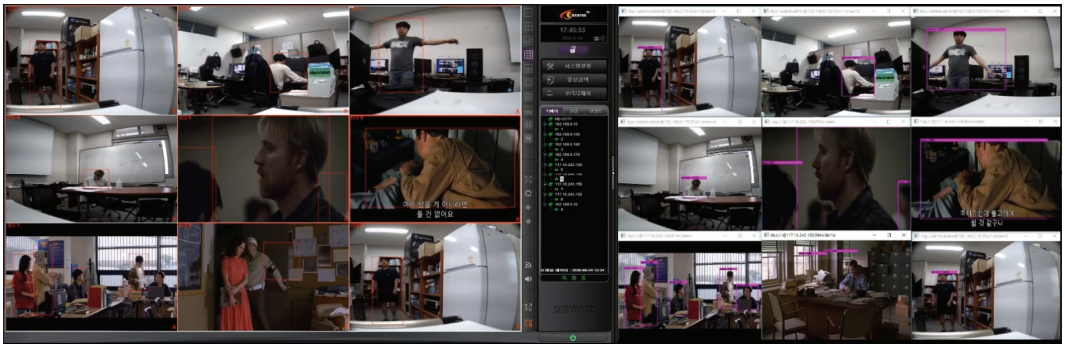
## 5.1 Experimental Environment

For the experiment, we evaluated the performance of RAVIP on two systems with different specifications, as shown in Table 2. For real-time video input, four IP cameras are connected by RTSP, and 12 different real-time videos are transmitted to RTP and RTSP through an external streaming server. To check the service of the back-end server, one NVR receives the same camera and video input as the server, and outputs the service result of the back-end server.

**Table 2.** Experimental system specifications

| | System A | System B |
|---|---|---|
| CPU | i7-8750H | i9-9900KF |
| GPU | GeForce GTX 1060 6GB | GeForce RTX 2080 TI |
| Memory | 16 GB (DDR4) | 32 GB (DDR4) |
| O/S | Windows10 | Windows10 |

Above all, the main goal of this study is whether it is possible to simultaneously support multiple object detection services for multi-channel CCTV input in a relatively low specification system. Fig. 7 shows the results of interworking the proposed RAVIP with the actual NVR. This experiment shows the actual results according to the NVR's nine service requests in the System B environment of Table 2.



(a) (b)

**Fig. 7.** NVR display serviced by RAVIP: (a) the processing screen in RAVIP server and (b) the screen receiving real-time object detection service from RAVIP in NVR.

## 5.2 Performance Evaluations

### 5.2.1 Multi-channel frame rates

In Fig. 7, the proposed RAVIP can provide real-time simultaneous object detection service from multi-channel inputs. In this experiment, the number of service channels that can be simultaneously supported on the same system and the performance of the existing YOLO and RAVIP for the FPS serviced are compared. The frame rate of the video used as an input in this experiment is 25 FPS, and RAVIP and YOLO were executed in System A and System B of Table 2, respectively, and the service execution result for the same service request was observed. Fig. 8 shows the comparison result of the actual service performance of RAVIP and YOLO.

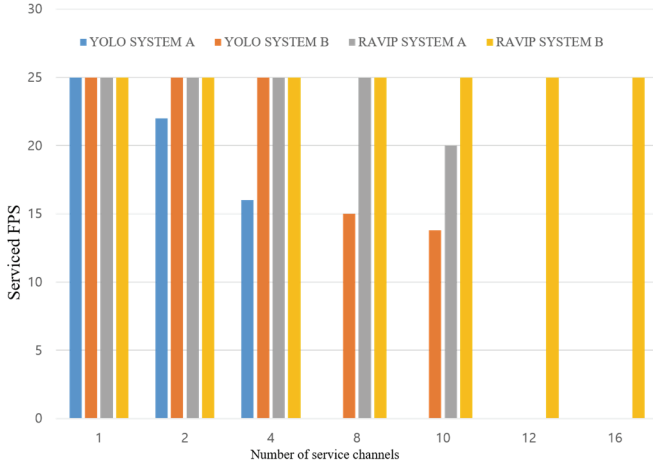



**Fig. 8.** Serviced FPS (YOLO vs. RAVIP).

As shown in the results, it is shown that in the case of the existing YOLO, only up to 10 simultaneous services can be performed even in System B, which is a relatively high specification. In addition, the FPS for the services for 8 and 10 channels are 15 and 13.8 FPS, respectively, and it can be seen that the performance of YOLO is rapidly degraded. In addition, it can be seen that in System A, which is a lower specification system, only up to 4 simultaneous channels can be serviced, and service is disabled in more channels. Service FPS can also maintain the same 25 FPS as the input channel only when a single service is supported, and in the service for 2 and 4 channels, the performance is rapidly degraded to 22 and 16 FPS, respectively.

That being said, in the case of the proposed RAVIP, it can be seen that up to 16 channels are available in System B. A more notable result is that it is possible to support high quality service of 25 FPS per-channel even when performing 16 channel services at the same time. The results of this experiment prove that the proposed RAVIP can serve at least 6 channels more than the existing YOLO on the same system, and the performance of the service at this time is not degraded.

### 5.2.2 Resource utilization

An important goal of the RAVIP service is to run multiple object detection modules simultaneously on streams received over multiple channels. The important thing here is how efficiently each instance can use limited hardware resources. Thus, this subsection evaluates the hardware resource usage of RAVIP in two different systems. In particular, we have identified CPU, GPU, and memory utilization in multiple channel services.

Fig. 9(a) shows the change in CPU utilization as the number of service channels increases. As shown in the Fig. 9(a), it can be seen that up to 4 channels serviced simultaneously, the service for each channel is performed within 15% of CPU utilization. This is because complex operations for real object detection are actually performed on the GPU. Still, when the number of channels further increases, utilization increases rapidly because the load in the network for receiving real-time frames from an input source per-channel increases. But System B can still maintain high FPS for each channel until CPU utilization reaches 98%.

Fig. 9(b) shows GPU utilization. Since the GPU is used intensively when the object detection module is executed, this result measured the peek value of GPU utilization. In this result, a remarkable feature is that in the case of System B, GPU resource usage does not show an exponential increase even with an increase in the number of channels compared to CPU utilization, and it was confirmed that only 16% of resources are used in 16 channels. This is in contrast to the experimental results in Fig. 9(a). In other words, RAVIP maximizes the use of GPU for complex calculations of the object detection module, and allows the CPU to take charge of other calculations, thereby expanding the number of service channels. In the case of System A, the reason that the slope for GPU utilization decreases from 10 channels is because the service FPS decreases as the number of channels increases, as shown in Fig. 8. Although the GPU utilization rate remains in the 50% range, the reason that services for additional channels are no longer possible is due to a problem arising from memory utilization.

In addition to the previous two experiments, we observed memory usage in the multi-channel object detection service as shown in Fig. 9(c). As shown in the result, it can be seen that in System A, when the number of service channels increases by 4 or more, it increases rapidly, and in the end, when 10 channels are serviced, 50% of the total memory is used. In actual O/S, since the individual process blocks more

than 50% of memory from being used, it can be seen that the maximum number of channels that can be processed in each system eventually depends on the memory usage. Nevertheless, unlike YOLO, which performs object detection as an individual process per-channel, as shown in Fig. 8, RAVIP, in which a single process processes object detection with multiple instances in parallel, can provide services for more channels at the same time.

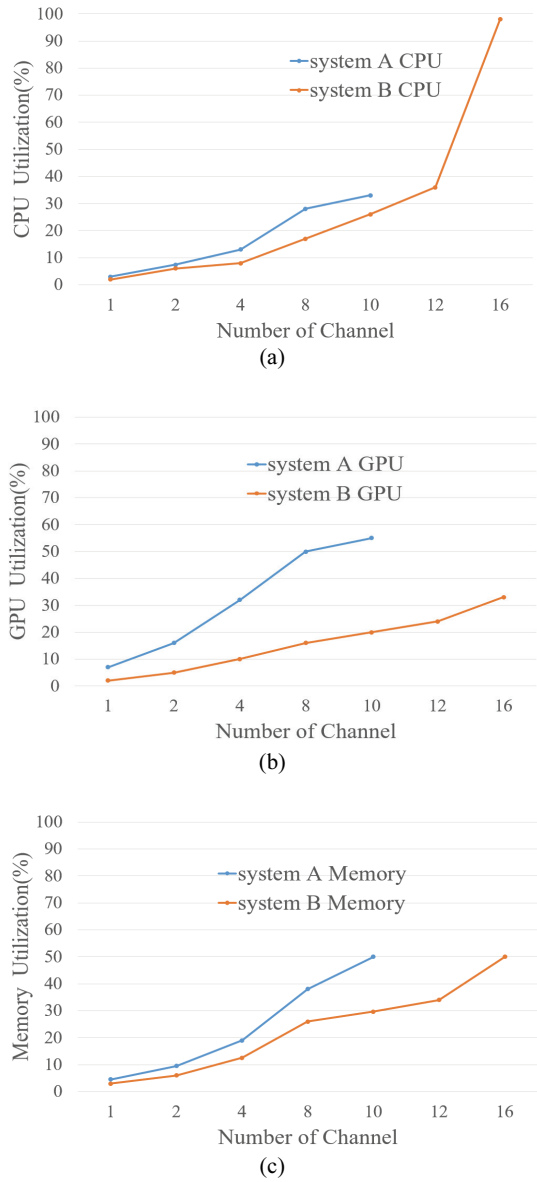



**Fig. 9.** Resource utilization of (a) CPU, (b) GPU, and (c) memory.

### 5.2.3 Delay analysis

In addition to the evaluation of resource utilization, we observed the processing speed of the object detection module and the transmission time of the processed results. First, processing delay is the time

taken to return the result after performing object detection by inserting one frame received from the fetch thread as input to object detector, and the measurement result is shown in Fig. 10(a). It can be seen that the processing delay increases as the number of channels increases by parallel processing on multiple channels at the same time. Nevertheless, it can be seen that System B shows an almost linear increase up to 12 channels at 7 ms or less, then increases rapidly in the 16 channels. This is because especially utilization of CPU and memory resources increases rapidly from that point.

In addition, we measured the time (metadata transmission time [MDTT]) taken to transmit the result of object detection processing to the NVR that requested the service. Fig. 10(b) shows the change of MDTT according to the simultaneous input channel increase. The notable feature here is that MDTT is not affected by the increase in channel and has a consistent transmission time (System A = 41 ms, System B = 47 ms) Will show This indicates that a number of back-channels for transmitting results can display consistent results because they are supported by the OS's TCP/IP STACK service.

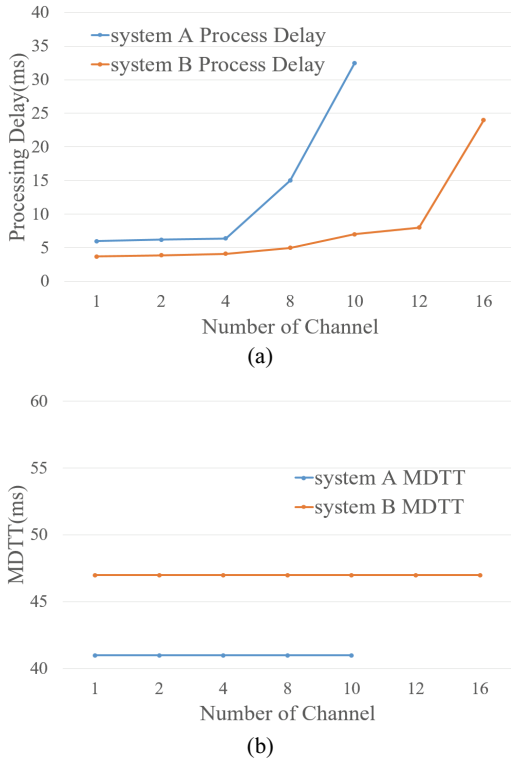



**Fig. 10.** Delay analysis: (a) processing delay time and (b) metadata transmission time.

## 5.3 Summary of Experimental Results

Through a number of experiments performed in Section 5.2, we have seen that, above all, RAVIP performs higher quality services in more channels than using multiple YOLO processes. In particular, in the case of RAVIP, it was confirmed that 16 channels can be simultaneously serviced in a system equipped with RTX 2080Ti GPU. This proves that the performance is improved by about 50% compared to YOLO, which only serviced up to 10 channels on the same hardware. In addition, through the observation of the service FPS for each channel, it can be seen that YOLO decreases as the number of

channels increases, while RAVIP shows that 25 FPS can be maintained even for simultaneous 16-channel service. Also, through experiments on RAVIP's own performance, we saw that RAVIP maximizes the utilization of GPU and CPU while being efficient in memory usage. In fact, in the case of System A, it was confirmed that while CPU utilization reached almost 98%, GPU and memory usage was reduced to less than 50%, enabling multi-channel services.

## 6. Conclusion

In this paper, we proposed RAVIP for local back end object detection service that can be associated with DVR or NVR. RAVIP solves the limitations and problems of object detection for existing multi-channel input sources, enabling efficient object detection services to be provided by DVRs and NVRs that monitor and store multi-channel CCTV. In particular, the creation and termination of modular per-channel instances based on RODEM Base maximized the utilization of back-end servers with limited resources. Through experiments on the back-end server with different hardware specifications, it was shown that the simultaneous object detection service of 16 channels can be serviced without any degradation or delay in recognition performance in the combination of RTX 208Ti based GPU and i9-9900 CPU. The biggest contribution of this study is to propose an efficient AI service structure for artificial intelligence back end services that can be linked with a myriad of currently deployed DVRs and NVRs, rather than a completely new independent system. NVR developers can easily develop services (intrusion detection, roaming, abandoned, etc.) without specialized knowledge of AI.

## Acknowledgement

This research was supported by Incheon National University Research Grant in 2018.

## References

[1] Wikipedia, "Digital video recorder," [Online]. Available: https://en.wikipedia.org/wiki/Digital_video_recorder.

[2] Wikipedia, "Network video recorder," [Online]. Available: https://en.wikipedia.org/wiki/Network_video_recorder.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA, 2005, pp. 886-893.

[4] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the 7th IEEE International Conference on Computer Vision*, Kerkyra, Greece, 1999, pp. 1150-1157.

[5] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Proceedings of the 6th International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, Bombay, India, 1998, pp. 555-562.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, 2014, pp. 580-587.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, 2016, pp. 779-788.

[8] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, 2017, pp. 7263-7271.

[9] J. Redmon and A. Farhadi, "Yolov3: an incremental improvement," 2018 [Online]. Available: https://arxiv.org/abs/1804.02767.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, 2017.

[11] J. Parsola, D. Gangodkar, and A. Mittal, "Post event investigation of multi-stream video data utilizing Hadoop cluster," *International Journal of Electrical & Computer Engineering*, vol. 8, no. 6, pp. 5089-5097, 2018.

[12] S. Karimi-Mansoub, R. Abri, and A. Yarici, "Concurrent real-time object detection on multiple live streams using optimization CPU and GPU resources in YOLOv3," in *Proceedings of the 4th International Conference on Advances in Signal, Image and Video Processing*, Athens, Greece, 2019, pp. 23-28.

[13] L. Tan, X. Dong, Y. Ma, and C. Yu, "A multiple object tracking algorithm based on YOLO detection," in *Proceedings of 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, Beijing, China, 2018, pp. 1-5.

[14] D. Kim and S. Park, "An intelligent collaboration framework between edge camera and video analysis system," in *Proceedings of 2018 International Conference on Electronics, Information, and Communication (ICEIC)*, Honolulu, HI, 2018, pp. 1-3.

[15] G. Mattela, C. Pal, M. Tripathi, R. Gavval, and A. Acharyya, "Enterprise class deep neural network architecture for recognizing objects and faces for surveillance systems," in *Proceedings of 2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, Bengaluru, India, 2019, pp. 607-612. IEEE.

[16] J. S. Park, M. Wiranegara, and G. Y. Son, "Multi-channel video analysis based on deep learning for video surveillance," *The Journal of the Korea Institute of Electronic Communication Sciences*, vol. 13, no. 6, pp. 1263-1268, 2018.

[17] Y. Wu, Z. Zhao, S. Zhang, L. Yao, Y. Yang, T. Z. Fu, and S. Winkler, "Interactive multi-camera soccer video analysis system," in *Proceedings of the 27th ACM International Conference on Multimedia*, Nice, France, 2019, pp. 1047-1049.

[18] S. Aslan and S. C. Ileri, "Performance analysis of ARM big.LITTLE architecture based mobile processor with multi-thread face detection," in *Proceedings of 2019 4th International Conference on Computer Science and Engineering (UBMK)*, Samsun, Turkey, 2019, pp. 336-339.

[19] X. Wang, "An efficient end-to-end object detection pipeline on GPU using CUDA," Master's thesis, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 2019.

[20] S. Liu, G. Liu, and H. Zhou, "A robust parallel object tracking method for illumination variations," *Mobile Networks and Applications*, vol. 24, no. 1, pp. 5-17, 2019.

[21] NVIDIA, "DeepStream SDK 5.1," 2021 [Online]. Available: https://developer.nvidia.com/deepstream-getting-started.

[22] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," Internet Engineering Task Force, Fremont, CA, RFC 3550, Standard 64, 2003.

[23] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," Internet Engineering Task Force, Fremont, CA, RFC 2326, 1998.

**Jeonghun Lee** **https://orcid.org/0000-0002-9617-2901**

He received B.S. degree in embedded system engineering from Incheon National University, Incheon, Korea, 2020. Since March 2020, he is with the embedded system engineering from Incheon National University as a master's course. His current research interests include embedded system, image processing system.

**Kwang-il Hwang** **https://orcid.org/0000-0002-4196-4725**

He is a full professor, in the department of embedded systems engineering, Incheon National University, Korea. He has received M.S. and Ph.D. degrees in Electronics and Computer Engineering from Korea University, Seoul, Korea, 2004 and 2007, respectively. His research interests include applied AI systems, IoT, and mobile robots. He is a member of IEEE, KICS, KMMS, KIPS, and KISS.