Towards Dynamically Reconfigurable IoT Camera Virtualization for Video Analytics Edge Cloud Services

Dongman Lee, Si Young Jang, Byoungheon Shin, and Yoonhyung Lee
Korea Advanced Institute of Science and Technology

***Abstract*—Video analytics edge computing exploiting IoT cameras has gained high attention. Running such tasks on the network edge is very challenging since video and image processing are bandwidth-hungry and computationally intensive. IoT cameras are heavily dependent on environmental factors such as the brightness of the view. In this paper, we propose an edge IoT camera virtualization architecture that enables an IoT camera to accommodate multiple application operation semantics and dynamically adjust its configuration to preserve them in the presence of environment context changes. For this, we develop an ontology-based application description model, a virtualization architecture with the container technology, and a context-aware dynamic reconfiguration scheme.**

Edge computing,[1] **also** referred to as cloudlet,[2] fog computing,[3] and MEC,[4] is an emerging technology that enables users to exploit mobile devices or Internet of Things (IoT) in close proximity. Recently, video analytics edge computing exploiting IoT cameras has gained high attention in solving many real-world problems.[5] However, since video and image processing are bandwidth-hungry and computationally intense, running such tasks on network edges is still very challenging.

Digital Object Identifier 10.1109/MIC.2019.2893872
Date of publication 18 January 2019; date of current version 7 October 2019.

Published by the IEEE Computer Society

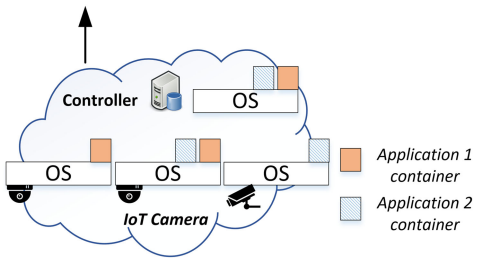

Figure 1. Camera edge IoT cloud.

Recent studies[6,7] adopt powerful edge servers (controllers) so that video analytic operations such as filtering, detection, and cropping are handled at the edge of the network. Other studies[8–11] show that less bandwidth utilization, higher accuracy, and energy efficiency can be achieved by coordinating multiple smart cameras via edge servers and transferring the only necessary information to a cloud or an application. In these systems, an edge controller acts as an intermediary between applications and edge IoT cameras. That is, the controller manages a set of cameras and provides the applications it serves with a higher level or virtualized view of underlying cameras and their capabilities, in other words, filtered video stream(s) according to an application's needs.

Unlike a traditional computing resource, an edge IoT camera is heavily dependent on context changes in its surrounding environment.[12] Furthermore, a subtle change in an environmental factor such as brightness leads to immediate service quality degradation. However, the existing solutions mentioned above fail to promptly recognize the environmental context changes, which are often critical to video analytics applications. This is because the controller can only validate the usefulness of the video frames after they are received from the cameras. The IoT camera records poor quality (or less-useful) frames until the controller makes adjustments to them. This motivates us to devise a new approach that enables an IoT camera to instantly recognize the context changes and accordingly adjust its configuration (e.g., streaming parameters) for applications that its controller serves while application-specific frame manipulation is continued to be done by the controller.

In this paper, we propose an edge IoT camera virtualization architecture (see Figure 1) that enables an IoT camera to be virtually configured to support multiple applications and dynamically adapt its configuration to environmental context changes for consistent QoS support for applications that it feeds video streams. For this, the proposed architecture features three key components:

1. An interpretation scheme for understanding the application requirements.
2. IoT camera virtualization to serve multiple video analytic applications.
3. Context-aware IoT camera reconfiguration to adapt to dynamic environmental context changes.

Our prototype shows that virtualization of multiple IoT cameras to serve applications and reconfigure the cameras in our proposed approach can significantly enhance object detection accuracy and reduce response time, which results in the improvement of application performance.

IOT CAMERA AS AN EDGE COMPUTING NODE

Motivating Scenario

In a public place such as an airport, there are various video analytics applications continuously and simultaneously running to ensure public safety. For example, a surveillance camera always runs a monitoring application, which detects suspicious actions. Along with the monitoring application, a suspect tracking application may also need to track down a suspect that requires a different set of parameters for camera operations (e.g., higher frame rate). Upon environmental context changes (e.g., a light is suddenly turned OFF), the camera switches to a different detection algorithm that is less affected by luminosity in order to support the required video frame rate without missing useful frames.

To realize such a scenario, the following is key design issues that we need to consider:

How to Support Application-Aware IoT Camera Resource Abstraction?

As shown in the scenario above, a single camera can run different applications: one for monitoring and the other for tracking. However, there exist common parameters such as the required frame rate that can be expressed in the same

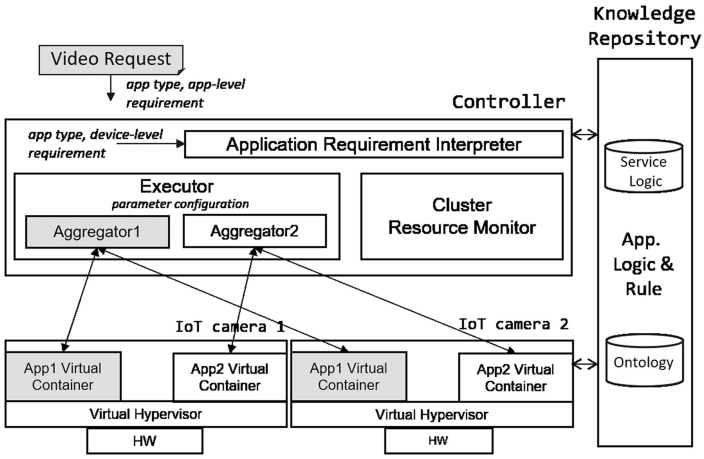

Figure 2. Camera virtualization architecture for an edge IoT camera cloud.

way because both are using a common video capability. This parameter can also affect the QoS of the corresponding application service, which also varies from application to application. For instance, if the IoT camera is not aware of these application requirements, the computational power on the device will be equally distributed among all applications. In this situation, the tracking application may suffer in performance from slower frame rate than required while the monitoring application does not. This requires a single physical IoT camera to host multiple virtual cameras where each virtual camera configures its setting according to the corresponding application.

To correlate the QoS requirements of a target application service type with the functionality of IoT devices, each IoT camera in a cloud should be semantically represented by the QoS requirements of each application service type that it will serve, not by its computing capability. For a video surveillance IoT, video recording fidelity factors such as resolution or brightness level of the scene should be considered. To represent the capability of an IoT device in an IoT camera cloud in terms of service functionality, ontology can be considered one of most plausible solutions[13] as it can semantically express the IoT resources.

How to Dynamically Reconfigure IoT Camera in Presence of Context Changes?

As IoT cameras are placed close to an edge cloud environment, dynamic contexts affect directly the performance of the video analytics applications. For instance, a tracking application is very sensitive to the luminosity degradation of a camera due to external factors such as shade or weather change. The controller should perform additional adjustment (e.g., brighten or sharpen images) to overcome video quality degradation. Additional bandwidth is required to send more frames to the application for these adjustments. However, if a camera is capable to directly deal with such local environment context changes and adjust its configuration to support video quality consistently, the controller can reduce the networking and processing cost due to handling extra video frames and perform its task more effectively.

However, as the number of choices increases, selecting an optimal operation semantics is challenging. Thus, we need to reduce the search space while selecting a proper operation semantics that is close to optimal. When finding a suboptimal choice, we should consider not only the estimated quality but also the amount of demanded resources of each operation semantics since an IoT device is still resource-constraint.

PROPOSED ARCHITECTURE

Overview

Based on abovementioned challenges, we propose a context-aware camera virtualization architecture for an edge IoT camera cloud, as shown in Figure 2. It has three main components: knowledge repository, controller, and IoT camera. An IoT Camera is a node which has both specific functionalities (i.e., video recording) and computing capability. The controller accepts video requests (VRs) from applications and converts them to camera configurations referencing the knowledge repository. The controller then selects a set of IoT cameras satisfying the converted service capability. Each IoT camera calculates service capability based on environmental contexts and their resource usage. In order to support multiple application requests, each IoT camera is virtualized by splitting the physical camera into multiple virtual cameras (i.e., application containers) using containers. Along with environmental context changes, the IoT camera promptly reconfigures

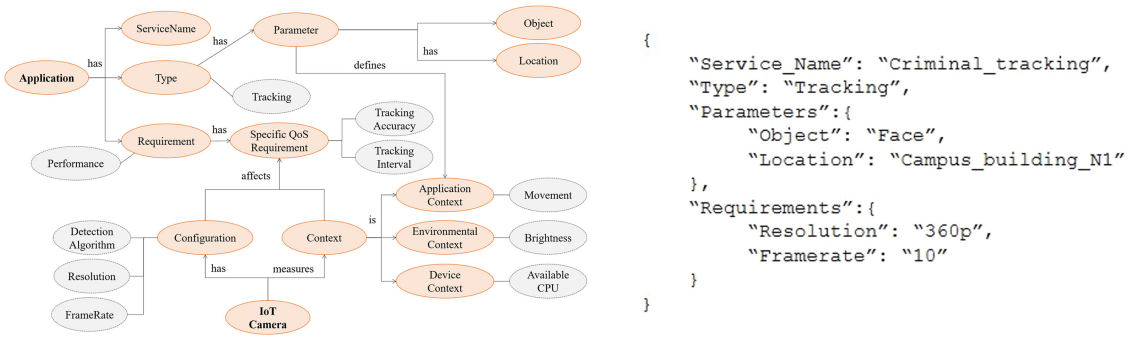

Figure 3. (a) Service ontology model and (b) video request example.

its parameters to continuously satisfy the application requirement. Following sections describe a detailed description of these processes.

Application-Aware IoT Camera Resource Abstraction

In order for an IoT Camera to process and filter video frames according to an application's needs, it has to understand the application's requirements. For expressing semantic relationships between concepts coming from different layers, ontology is considered a plausible solution.[13] Thus, we devise ontology-based relationships among an application, its QoS-affecting parameters, and environmental contexts. Both controller and IoT camera reference them via the knowledge repository.

Knowledge Repository The knowledge repository is a shared knowledge base between applications and an edge IoT camera cloud. It consists of a service ontology and a service logic database. The service ontology defines conceptual relationships between application requirements, camera configuration, and contexts. The knowledge repository is a separate entity from the controller and IoT cameras so that edge cloud administrators, IoT manufacturers, or cloud service developers can update knowledge instances and service logics without recompiling the controller and IoT cameras. For each application type, its logic representing the relationship between the QoS, contexts, and the configuration is built in the knowledge repository. Application logic is an instance of the ontology model, and the example of tracking applications is shown in Figure 3(a).

Video Request Each application sends a VR to the controller through an application programming interface. A VR describes the abstract features and requirements for utilizing IoT cameras using key-value pairs as shown in Figure 3(b). We use a JSON format since it is intuitive and widely used. All description formats stem from the keywords of the service ontology in the knowledge repository, and the scope of values are also limited by predefined instances in the ontology. Since existing edge computing approaches specify QoS requirements related to computation resources, the VR description extends the QoS requirements by decoupling application requirements from video-specific configurations. A VR consists of the VR name, a service type, filtering parameters, and application requirements. The service type helps the controller to initialize corresponding IoT cameras with default values from the knowledge repository. The executor sends a start command to serve video application containers to IoT cameras.

IoT Camera Virtualization

In order for an IoT camera to serve multiple video surveillance applications, we leverage docker, a container-based virtualization technique instead of virtual machines as it is a feasible solution in resource-constrained devices for quick allocation.[14] An IoT camera contains multiple virtual camera containers (VCCs) and a virtual camera hypervisor (VCH) as shown in Figure 4.

Virtual Camera Container A VCC is a docker container for serving a target application. It is created based on service capabilities converted

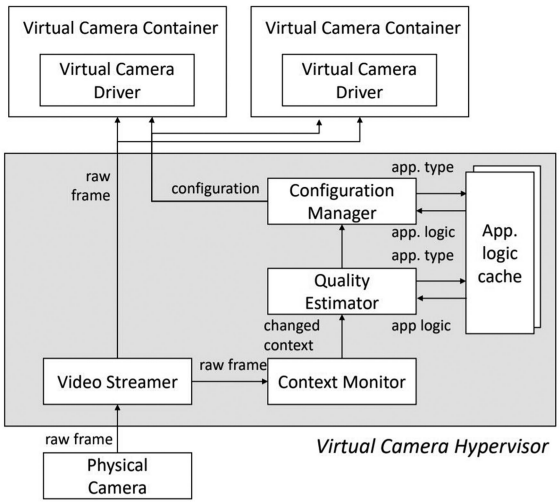

Figure 4. VCC and hypervisor.

from a VR by the controller. It runs application-specific processes such as a human detection operation. The process is supported by the virtual camera driver, a device driver for polling video frames from the physical camera according to application requirements. In other words, the virtual camera driver expresses an application-specific camera that is not physically present but within the capability limits of a physical camera hardware. There can be as many virtual drivers created as the number of applications running on the IoT camera.

Virtual Camera Hypervisor To virtualize an IoT camera for supporting multiple applications, we devise VCH. It dynamically configures a VCC according to the application QoS requirement and context changes. It has four modules: video streamer (VS), context monitor (CTM), quality estimator (QE), and configuration manager (CFM).

VS. It is directly connected to a physical camera and a VCC to mediate those two entities. It retrieves video frames from the physical camera and sends to the VCC. In the VS, the camera configuration is set with the highest quality to satisfy any application requirement.

CTM. It periodically captures contextual information from video frames obtained by the VS. It only monitors environmental contexts that can affect application performance. It also checks if the environmental context has an impact on the application QoS by performing simple image processing operations that do not consume an excessive amount of computation (e.g., calculating the difference in pixel values, or averaging them). It caches the lastly captured context values to avoid duplicate measurements by multiple applications requesting the same type of context. The cached contexts are also used to calculate the estimated quality. Whenever there is a change in the environmental context, the QE gets notified.

QE. It estimates the QoS of applications running on the IoT camera based on the environmental context changes. It refers to application logic in the knowledge repository to calculate the quality for each application, and application logic can be locally cached in the VCH for reuse. The application logic indicates an available range of environmental contexts to support a specific application requirement. If the estimated QoS is out of the required QoS range or the difference with the previous estimated QoS surpasses a threshold, the QE notifies to the CFM to reconfigure the VCC.

CFM. The CFM adjusts the configuration of a VCC according to the QoS requirement of a target application. One natural approach when selecting configuration is to heuristically apply configuration depending on the environmental context and camera configuration (e.g., processing algorithms). A new configuration is made to guarantee that it leads to a resulting QoS residing within the requirements of a target application while consuming the minimum amount of computation. For this, the *configuration selection algorithm* shown below iterates through possible configuration sets removes any set consuming more resources than the current set and chooses the top configuration. Then, this configuration is applied to the VCC.

```
Configuration Selection
{
  for each configuration set s_i in S
    q ← estimate quality (s_i, context)
    e ← estimate energy usage (s_i)
  if q > current QoS
    S.append (s_i)
  Sort S by q
}
```

In addition, since two or more application video streams must be served, to avoid conflict we prioritize the application container by type,

request time, and request entity. When applications with different priorities are executed at the same time, the configurations of all running applications should be balanced. For this, we define global quality (GQ) for each application $i \in I$ where I is a set of currently running applications. The GQ is expressed as follows:

$$GQ = \sum_{i \in I} w_i Q_i, \sum_{i \in I} w_i = 1$$

where w_i and Q_i are the weight and the quality of the application i, respectively. Each weight is set by the relative priority among applications. There is a threshold which is the lower limit of GQ. The threshold is set to a sum of values which are multiplied by each minimum required quality and priority. When GQ drops under threshold, the CFM restore the value by reconfiguring the VCC. To effectively maintain the GQ, the CFM improves the configuration of the application having higher weight and degrades the QoS of the application which has lower weight.

EVALUATION

We implement a proof-of-concept system of our architecture with multiple IoT cameras using Python 3.6. For dynamic context adaptation evaluation, we conduct two experiments to see how meaningful it is to change the configuration of an IoT camera depending on the environment, and how adaptive the camera itself is at recognizing the context and performing reconfiguration.

Experimental Setup

Controller and IoT Camera—We implement IoT cameras using Raspberry Pi 3 equipped with camera modules and the controller using a desktop computer. The controller can be flexibly implemented on a local server with more computing power. All nodes in our prototype testbed are wireless, thus transmit packets through built in WiFi (IEEE 802.11n). We utilize MQTT, a lightweight messaging protocol, for communicating between the controller and IoT cameras.

Experiment Scenario and Dataset—To evaluate our system, we prototype a tracking application. It tracks a person by continuously detecting a human figure and sends the cropped frame of the detected human to the controller. We use Haar cascade and HOG algorithms which are reasonably lightweight and thus suitable for Raspberry Pi. Our video set contains human in hallway environment.

Context-Aware IoT Camera Reconfiguration

Impact of Dynamic Reconfiguration—In order to evaluate the performance of adaptive reconfiguration, we run a tracking application using three different schemes: 1) a policy applying a fixed high-quality configuration; 2) a policy applying a fixed low-quality configuration; and 3) a context-aware adaptive reconfiguration policy. Low-quality configuration sets frame resolution to 400×300 pixels and runs the Haar cascade algorithm that is a relatively simple detection algorithm. On the other hand, high-quality configuration sets frame resolution to 600×450 pixels and runs the HOG algorithm which has a high recognition rate and consumes $1.5\times$ more computing power than the Haar cascade algorithm.

In our dynamic reconfiguration scheme, before the lighting setting change, the IoT camera uses the low-quality configuration (400×300, haar cascade) to detect. As its environment gets darker, it dynamically switches to the high-quality configuration (600×450, HOG) to avoid quality drop since the low-quality configuration may not recognize the change. Figure 5(a) and (b) clearly shows this—the static low-quality configuration cannot detect a human whereas the dynamic reconfiguration scheme can. As the situation gets normal, it switches back to the low-quality configuration. This allows our dynamic scheme to support a consistent detection rate, compared to the schemes with the static configuration as shown in Figure 5(c).

Adaptation Time for Context-Aware Reconfiguration—We evaluate how much time the proposed scheme reduces the reconfigure time when the context changes, compared with controller-based approaches.[8,9]

Figure 5(d) shows changes in the normalized quality of a tracking application, which refers to a normalized moving average value of correctly detected objects over a given time. The light is turned off at 13 seconds and turned on again at 24 seconds. The quality of the application drops in all cases at 13 seconds into the experiment.

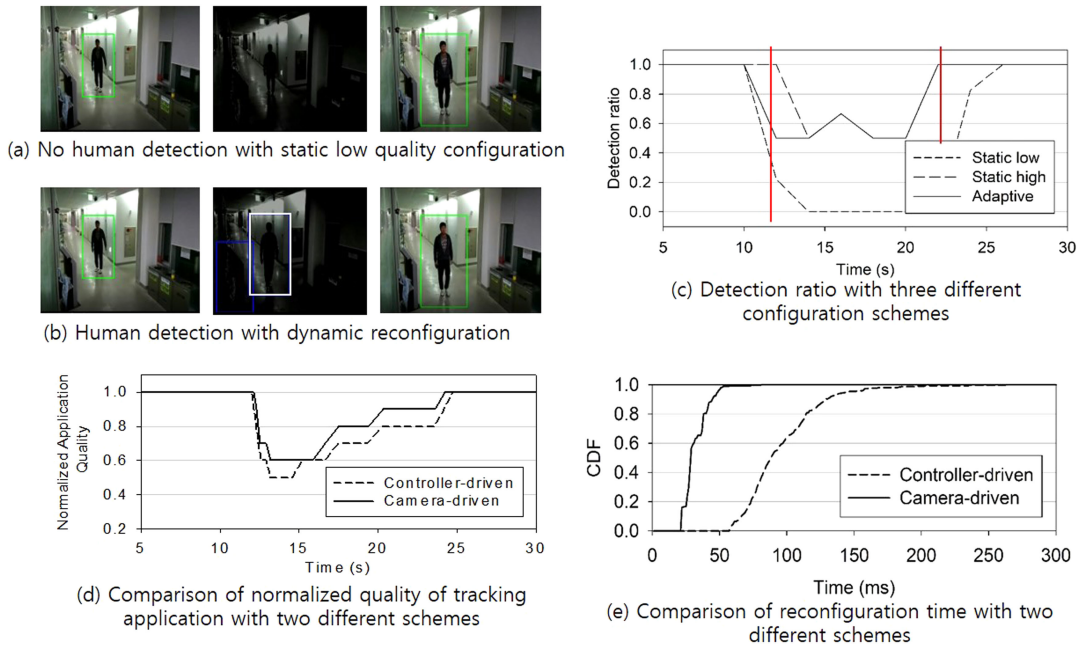

Figure 5. Evaluation results.

The controller-based approach shows that the system reverts (0.1 sec) back to a normal value because only metadata (brightness) is sent for evaluation. The quality of detection ratio in the proposed approach recovers faster (0.025 sec) as reconfiguration is done on the camera as well. Throughout the context change duration, the controller-based approach fails to accurately detect for approximately two seconds due to transmission of context values over a network.

Figure 5(e) shows the CDF of processing time for both approaches. A dashed line is the result of the controller-based approach and a solid line is the result of the proposed one. As shown in the figure, reconfiguration takes about 58 ms to 197 ms for the controller-based approach and 22 ms to 80 ms for the proposed approach. This is due to the transmission delay of frames through the network, which is not relevant in the proposed approach. This delay may cause approximately five video frames to drop depending on the cropped frame size. In summary, the proposed scheme shows up to 2.8 times shorter response time than the controller-based approach.

CONCLUSION

In this paper, we proposed an edge IoT camera virtualization architecture for video analytics applications. In order to enable an edge IoT camera to understand application requirements and environmental contexts, we define an ontology-based knowledge base with relationships between application features and IoT camera parameters. We leverage a container based virtualization technique that enables an IoT camera to support multiple application operation semantics. To dynamically adjust operation semantics according to environment context changes, we devise a context-aware reconfiguration scheme. The experimental results with our prototype implementation verify that the proposed architecture allows edge IoT cameras to effectively adapt to environment changes.

Our future work includes the following points. 1) We plan to leverage reinforcement learning such as Q-learning for more efficient reconfiguration upon context changes. Contexts from video frames can be modeled as a state space, and an optimal set of IoT camera configurations for given states can be learned at runtime based on the feedback from the controller such as detection rate and detection accuracy. 2) We will consider the cases where multiple IoT cameras are sequentially involved. For instance, as a user moves away from the view of a camera in charge, if user tracking is instantly taken over by another camera that can handle the same job in the user's moving direction, the quality degradation can be

minimized. It needs an IoT camera to be aware of other nearby cameras, find an alternative IoT camera, and transfer its responsibility to a new one.

ACKNOWLEDGMENT

This work was supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) funded by the Korean government (MSIT) under Grant 2016-0-00160 "Versatile Network System Architecture for Muti-dimensional Diversity" and Grant 2019-0-01126 "Self-learning based Autonomic IoT Edge Computing."

REFERENCES

1. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 23, no. 40, pp. 637–646, Oct. 2016.
2. M. Satyanarayanan *et al.*, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct.–Dec. 2009.
3. F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. 1st Edition MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.
4. Y. C. Hu *et al.*, "Mobile edge computing—A key technology towards 5G," ETSI white paper 11, vol. 11, pp. 1–16, 2015.
5. F. Loewenherz, V. Bahl, and Y. Wang, "Video analytics towards vision zero," *Inst. Transp. Eng. ITE J.*, vol. 87, no. 3, p. 25, 2017.
6. H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *NSDI*, 2017.
7. J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: Scalable adaptation of video analytics," in *Proc. Conf. ACM Special Interest Group*, 2018, pp. 253–266.
8. T. Zhang, A. Chowdhery, P. V. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," in *Proc. ACM MobiCom*, 2015, pp. 426–438.
9. T. Dao, K. Khalil, A. K. Roy-Chowdhury, S. V. Krishnamurthy, and L. Kaplan, "Energy efficient object detection in camera sensor networks," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 1208–1218.
10. T. Dao, A. Roy-Chowdhury, N. Nasrabadi, S. V. Krishnamurthy, P. Mohapatra, and L. M. Kaplan, "Accurate and timely situation awareness retrieval from a bandwidth constrained camera network," in *Proc. IEEE 14th Int. Conf. Mobile Ad Hoc Sensor Syst.*, 2017, pp. 416–425.
11. S. Maheshwari *et al.*, "Scalability and performance evaluation of edge cloud systems for latency constrained applications," in *Proc. IEEE/ACM Symp. Edge Comput.*, 2018, pp. 286–299.
12. S. Y. Jang, H. Choi, Y. Lee, B. Shin, and D. Lee, "Semantic virtualization for edge-IoT cloud: Issues and challenges," in *Proc. 2nd Workshop Cloud-Assisted Netw.*, 2017, pp. 55–60.
13. A. I. Maarala, X. Su, and J. Riekki, "Semantic reasoning for context-aware Internet of Things applications," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 461–473, Apr. 2017.
14. R. Morabito, "Virtualization on internet of things edge devices with container technologies: A performance evaluation," *IEEE Access*, vol. 5, pp. 8835–8850, 2017.

Dongman Lee is a Professor with the School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, South Korea. His research interests include smart space middleware, edge IoT virtualization, social media analysis, and trust management. He is a member of KISS and IEEE, and a Senior Member of ACM. Contact him at dlee@kaist.ac.kr.

Si Young Jang is currently working toward the Ph.D. degree with the School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, South Korea. His current research interests include edge computing, edge IoT virtualization, and wireless networking. Contact him at sy.jang@kaist.ac.kr.

Byoungheon Shin is currently working toward the Ph.D. degree with the School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, South Korea. His research interests include wireless networking, edge computing, and social media analysis. Contact him at bhshin@kaist.ac.kr.

Yoonhyung Lee is currently with Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. Her current research interests include IoT framework and edge computing. She received the M.Sc. degree with the School of Computing, KAIST. Contact her at nyongee16@kaist.ac.kr.