https://doi.org/10.20965/jrm.2021.p0686

Paper:

Vision-Based Sensing Systems for Autonomous Driving: Centralized or Decentralized?

Manato Hirabayashi*, Yukihiro Saito, Kosuke Murakami**, Akihito Ohsato**, Shinpei Kato**,***, and Masato Edahiro***

*Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, Aichi 464-8603, Japan
E-mail: hirabayashi@ertl.jp
**Tier IV, Inc.
Jacom Building, 1-12-10 Kitashinagawa, Shinagawa-ku, Tokyo 140-0001, Japan
***Graduate School of Information Science and Technology, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
[Received January 25, 2021; accepted April 15, 2021]

The perception of the surrounding circumstances is an essential task for fully autonomous driving systems, but its high computational and network loads typically impede a single host machine from taking charge of the systems. Decentralized processing is a candidate to decrease such loads; however, it has not been clear that this approach fulfills the requirements of onboard systems, including low latency and low power consumption. Embedded oriented graphics processing units (GPUs) are attracting great interest because they provide massively parallel computation capacity with lower power consumption compared to traditional GPUs. This study explored the effects of decentralized processing on autonomous driving using embedded oriented GPUs as decentralized units. We implemented a prototype system that off-loaded image-based object detection tasks onto embedded oriented GPUs to clarify the effects of decentralized processing. The results of experimental evaluation demonstrated that decentralized processing and network quantization achieved approximately 27 ms delay between the feeding of an image and the arrival of detection results to the host as well as approximately 7 W power consumption on each GPU and network load degradation in orders of magnitude. Judging from these results, we concluded that decentralized processing could be a promising approach to decrease processing latency, network load, and power consumption toward the deployment of autonomous driving systems.

Keywords: decentralized processing, autonomous vehicles, graphics processing unit (GPU), image processing

1. Introduction

Autonomous driving is attracting considerable interest owing to its potential as a promising solution to transportation problems, including serious traffic accidents and declining quality of life caused by the lack of means for personal transportation in aging societies. Typically, autonomous driving systems mainly consist of three kinds of tasks: perception, planning, and control. Society of Automotive Engineers (SAE) International in the U.S.A. published "Levels of Driving Automation" (SAE J3016) [1] in 2016, and various counterparts have been defined in many nations (e.g., Society of Automotive Engineers of Japan (JSAE) established "Taxonomy and definitions for terms related to driving automation systems for On-Road Motor Vehicles" [2] in 2018). In such definitions, systems classified into level three and above primarily account for driving operations. Perception tasks receive data captured by external sensors, including cameras as well as light detection and ranging (LiDAR) units, as input and perform semantic understanding regarding the surrounding circumstances of ego vehicles. Although typical perception tasks are computationally intensive and receive large quantities of data as input, the long processing time for them is not tolerated by systems classified into level three and above, because the output of the tasks becomes input for following planning and control tasks. When considering the actual deployment of autonomous driving systems, a vehicle mounted system that performs online processing has severe limitations, including low power and computation resource consumption. Perception tasks suffer from these limitations, and the trade-off between the limitation and processing speed is concerned. Moreover, the system scaling should also be concerned. To ensure the safety of autonomous driving by sensing the surrounding circumstances of ego vehicles, multiple external sensors should be used. As the number of external sensors increases to eliminate blind spots, data quantity to be processed increases drastically, and the required computation resources increase as well. To realize safety autonomous driving systems, tolerance against this data quantity scaling is essential.

Embedded oriented graphics processing units (GPUs), including Jetson series [a–d] supplied by NVIDIA Corp. and Mali series [e] designed by ARM Ltd., have received

Table 1. Comparison of several Jetson series.[1]

	TX1	TX2	AGX Xavier	Nano
Release	2016	2017	2018	2019
Number of GPU core	256	256	512	128
Number of tensor cores	–	–	64	–
GPU architecture	Maxwell	Pascal	Volta	Maxwell
Memory amount	4 GB	8 GB	16 GB	4 GB
Power	Under 10 W	7.5 W / 15 W	10 W / 15 W / 30 W	5 W / 10 W
Deep learning accelerator	–	–	2x NVDLA Engines	–

much attention because they provide massively parallel computation capacity with lower power consumption compared to traditional GPUs. In particular, Jetson series have been developed as embedded platforms for autonomous machines. Although Jetson series are powerful devices, no single Jetson is supposed to be sufficient to deal with all tasks composing autonomous driving systems, because huge quantities of data and computational resources are required for the whole system and they may exceed the capacity of a single Jetson. However, data and computation resources that should be consumed by a host machine can be suppressed by constructing a decentralized system that off-loads parts of perception tasks onto edge devices, such as Jetson, and feeds solely processed results to the host. Although there have been works [3–6] regarding the processing performance on edge devices, they are focused solely on processing time, which is a delay on the internal edge (i.e., many studies tend to focus on non-system-wide performance). Therefore, there is still need for discussing problems that may occur when introducing decentralized processing into autonomous driving, including the delay caused by data transfer, the throughput of the whole system, and the tolerance against system scaling.

To this end, we constructed a prototype of a decentralized processing system that off-loaded parts of perception tasks for autonomous driving onto edge devices and explored the processing performance of the whole system, including the delay caused by introducing decentralized processing. Based on evaluation results, we discussed the validity of decentralized processing for autonomous driving systems. As mentioned above, typical autonomous driving systems consist of perception, planning, and control tasks. If the computational burden of all tasks concentrates on a specific unit (i.e., centralized unit) in a system, it may degrade the whole system throughput that causes serious traffic accidents in the autonomous driving context. Especially, typical perception tasks tend to consume a large amount of computational resources since they are computationally intensive. Therefore, the main aim of our decentralized processing system is to avoid concentrating computational burden to a specific location in a whole autonomous driving system.

The main contributions of our study are:

- We measured the system-wide-delay of an object detection task with and without decentralized processing and discussed the validity of the decentralized processing for the perception tasks of autonomous driving systems.
- We measured various factors that are concerned for autonomous driving systems, including the resource consumption and the processing speed of the object detection task on edge devices.
- We evaluated the effect of network quantization on multiple embedded oriented GPU series. The results demonstrated that network quantization with lower operation precision did not always achieve better performance in some cases.

This study makes a novel contribution to the literature by exploring the benefits and effects of introducing decentralized processing to autonomous driving systems by constructing a prototype system. We believe that the presented results could be helpful for considering decentralized processing on the actual deployment of autonomous driving systems.

The remainder of this paper is organized as follows. Section 2 examines previous related research on applications using Jetson. Section 3 describes the system model and assumptions of this study. Section 4 presents the implementation details, including the implementation platforms and techniques for network quantization, of this study. The evaluation regarding the above contributions is discussed in Section 5, and the conclusion and suggestions for future work are presented in Section 6.

2. Related Works

NVIDIA Corp. provides the Jetson series to meet various requirements, including computing performance and power consumption limitation. **Table 1** shows the comparison of the specification summaries of several Jetson series. The Jetson series have attracted attention as devices that accomplish high computing performance and

1. Partially cited from https://developer.nvidia.com/embedded/develop/hardware [Accessed January 24, 2021] and modified.

low power consumption simultaneously, and some studies have been published.

In [4], the authors studied porting deep learning (DL) approaches to a small and power efficient device. They focused on pedestrian detection using DL and analyzed the suitability of the Jetson TX1 mounted on a mobile robot in comparison with a high performance GPU. They also investigated the effect of changing the operation precision utilized in the convolutional neural network (CNN) for pedestrian detection from typical 32-bit floating point to 16-bit floating point, thereby resulting in 15 times acceleration on processing. [5] proposed an unmanned aerial vehicle warning system using onboard and real-time object detection. To fulfill the requirements of the system, such as minimal power consumption, limited onboard processing power, and minimal weight, the authors of [5] employed Jetson TX2. The authors of [6] evaluated various algorithms of *visual simultaneous localization and mapping*, which estimate the self-localization of mobile robots by using Jetson TX2. They reported that, by employing the embedded GPU, the CPU load decreases and the processing speed improves.

Including these related works, many evaluation results and onboard processing systems for perception tasks using Jetsons have been reported. Many of them focus on the processing time of a single task executed on a Jetson. In other words, they treat embedded oriented GPU units as the host processors of the systems, not edge devices. As mentioned in Section 1, such units are not sufficient enough to be the hosts of fully autonomous driving systems. However, we may leverage embedded oriented GPU units when applying them to autonomous driving systems as edge devices, whereas the influence regarding system integration, including the delay caused by introducing them as decentralized processing platforms, remains unclear. Furthermore, few previous works implementing Jetson AGX Xavier have been conducted because it is an emerging device relative to the other Jetson series. In this study, we explored the influence of introducing Jetson AGX Xavier as a decentralized processing platform and discussed the validity of decentralized processing on perception for autonomous driving systems.

3. Model and Assumptions

In this section, we present the model and assumptions of our study. We consider it is improper to apply decentralized processing to all perception tasks that arise in autonomous driving systems since its validity depends on processing contents. The target types of perception tasks in this study are assumed to have some attributes: "all direction sensing" and "tasks divisible into subtasks," which are frequently emerged ones in the context of autonomous driving. This is because we employed an architecture that divided perception tasks for surrounding circumstances (i.e., all direction) into subtasks and off-loaded them onto decentralized units.

As a target of evaluation, we assumed a model that consisted of one host PC and multiple Jetsons connected to cameras individually. Every Jetson performed object detection on images captured by the connected cameras, and all the detection results were concentrated at the host PC via network. **Fig. 1** depicts an overview of the model. As shown in **Fig. 1**, the host PC is connected to the Jetsons via a hub, and every Jetson is solely in charge of processing for the one specified camera. We consider this configuration is a typical one for a decentralized processing system for perception. By evaluating the performance of this configuration, we explore the validity of decentralized processing. Image-based object detection is one of the perception tasks that are suited to acceleration using GPU. One-stage detectors presented in recent years, including Single Shot MultiBox Detector (SSD) [7] and You Look Only Once (YOLO)v3 [8], perform bounding box proposal and classification in one network and make an object detection task run in almost real time. These CNNs contain computationally intensive operations, and the massively parallel computation capability of GPU underlays the fast processing of such operations. We employed YOLOv3-416 [8] written in TensorRT [f] as an object detection task that is executed on every Jetson. For decentralized processing, it would be common to assign tasks according to the processing capacity and network bandwidth of each decentralized processing unit. However, if we divided the perception of surrounding circumstances into subtasks perceiving a small region and afterward distributed those subtasks onto decentralized units, each unit would theoretically own equal load. Additionally, the processing capacity and network bandwidth of each decentralized unit should be equal since we assumed uniform series (i.e., Jetson) as the units. Hence, in our model, we distributed uniform subtasks of perception onto all decentralized units and did not consider complicated task assignments.

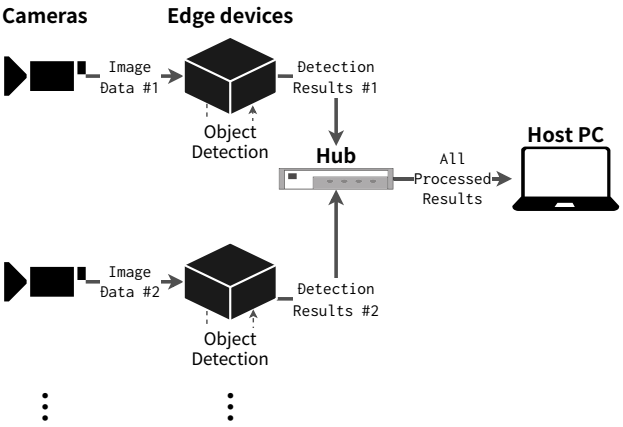

Fig. 1. System model of our study.

We utilized Jetson AGX Xavier [c] as decentralized processing platforms. The Jetson series consists of the CPU, GPU, dynamic random access memory (DRAM), and power management integrated circuit (PMIC), and they are intended to run various applications fast with

Table 2. Part of the power mode preset for Jetson AGX Xavier.[2] Mode "15 W" is the factory default.

Property	Mode name		
	EDP	10 W	15 W
Power budget	n/a	10 W	15 W
Mode ID	0	1	2
Online CPU	8	2	4
CPU max freq. [MHz]	2265.6	1200	1200
GPU TPC[3]	4	2	4
GPU max freq. [MHz]	1377	520	670
DLA[4] cores	2	2	2
DLA max freq. [MHz]	1395.2	550	750
VA[5] cores	2	0	1
VA max freq. [MHz]	1088	0	550
Memory max freq. [MHz]	2133	1066	1333

low power consumption. In addition to the massively parallel computation capacity of GPUs, the combination with TensorRT [f] is effective, especially to accelerate inference processing with deep neural networks (DNNs). TensorRT is a software development kit (SDK) that is provided by NVIDIA Corp. and maximizes the throughput using *network quantization*, which is an emerging technique to accelerate the inference speed of DNNs. The Jetson series can configure system settings, including the operating frequency and the number of online cores, through software. Although users can set their values for those configurations, some presets are available; this function is called *power mode*. These presets are useful to balance between computation power and power consumption, because the power consumed by Jetson can be roughly estimated according to each mode. This estimated power consumption is also referred to as *power budget*. Seven presets are available for Jetson AGX Xavier, and **Table 2** presents some of them.

4. Implementation

This section presents schemes for the decentralized processing system. First, we describe the platforms and their features to build the decentralized processing system. Second, we explain the techniques regarding DL inference on GPUs to provide how we accelerate inference to achieve real-time object detection. The rest of this section presents the configurations of the decentralized processing platform to explore the performance of the system.

2. Cited from https://github.com/dusty-nv/jetson-presentations/blob/master/20181004_Jetson_AGX_Xavier_New_Era_Autonomous_Machines.pdf [Accessed January 24, 2021]
3. TPC: Texture Processor Cluster.
4. DLA: Deep Learning Accelerator.
5. VA: Vision Accelerator.

4.1. Robot Operating System (ROS) Implementation

We deployed ROS [9] and Autoware [10, g] as platforms to implement the decentralized processing system. ROS is a middleware that provides useful tools and libraries to develop robot applications. It is well suited to decentralized processing because it employs a publisher-subscriber model to implement interprocess communication. Owing to this characteristic, functions consisting of a system such as camera drivers and object detectors can be developed as modules and easily connected to each other to work cooperatively. Autoware is open-source software for autonomous driving based on ROS. It contains modules that implement functions for perception, planning, and control for autonomous driving, including driver programs for various types of cameras. We employed a camera driver module provided by Autoware to acquire images from each camera.

In the field of ROS, the term "node" refers to an individual component module that constructs a system and the term "topic" refers to actual data that are *published* (i.e., transmitted) by nodes. Users can attach a timestamp to a topic programmatically. In our proposed system, camera drivers attach the current time at every publishing to image topics. Object detection nodes inherit the timestamp of *subscribed* (i.e., received) image topics and attach them to the corresponding output topics. To notice the communication and processing delays, we compared the timestamps of the topic published by the detection nodes and the instantaneous times at which an evaluation node received the topics. By subtracting these timestamps from the receiving times, we obtained the delays from image acquisition to the arrival of detection result to the host. In addition, to synchronize time between the host and the Jetsons, we installed a network time protocol (NTP) server on the host and NTP clients on each Jetson.

4.2. DL Inference Using TensorRT and Network Quantization

As mentioned in Section 3, we employed YOLOv3 written in TensorRT for the proposed system. TensorRT is an SDK for high performance DL inference provided by NVIDIA Corp. To improve inference performance, this SDK adjusts existing trained models against GPUs on which the models will be run. Changing operation precision is one of the adjustments performed by TensorRT. Although all the calculations on inference are performed in single float precision (FP32) by default, some of the calculations can be replaced by half float precision (FP16) or 8-bit integer precision (INT8) by explicitly specifying on execution. If operation precision is changed to FP16 or INT8, TensorRT generates an inference engine adjusted for the specified precision. By using the generated engine for inference, it is expected to improve the throughput and decrease the resource consumption [f]. Since NVIDIA Corp. has recently released some GPUs that contain exclusive cores to process INT8 operations, the operation throughput may be significantly improved and the

resource consumption may be degraded solely by shifting the operation precision to INT8. Because the available value range and granularity are quite different between single float precision and 8-bit integer precision, *calibration* is required to generate an INT8 inference engine. In the calibration process, TensorRT executes an FP32 inference engine for multiple inputs to explore the value range that the FP32 engine has actually taken internally and scales the range value into the 8-bit integer range. We also evaluated the amount of resource consumption and detection performance in each case of using FP32 and INT8 inference engines. To generate the INT8 inference engine, we used the 2017 validation image set [h] in MS COCO [11] for calibration.

4.3. Jetson Settings

To explore the potential capability of our system, we measured the minimum delays that could be achieved when Jetsons ran at full capacity. Additionally, we measured pure changes in power consumption when the operation precision was changed. To this end, we set the power budget of Jetsons to "n/a" (i.e., mode name "EDP" in **Table 2**). We will demonstrate the effects of power budget settings on the performance of the object detection task in Section 5.5.

5. Evaluation

To quantify the performance of the proposed system, which is a decentralized processing system for the environment perception of autonomous driving, we evaluated the following items:

1. Delays depending on the presence of decentralized processing.
2. Effects of network quantization on resource and power consumption.
3. Limitation of centralized processing.
4. Effects of network quantization on detection performance using different series of the edge device.

5.1. Experimental Setup

We deployed the following setups to evaluate our system.

Host PC (centralized environment)

- CPU: Intel core i7-8750H @ 2.20 GHz (12 core / 24 thread)
- Memory: 16 GB
- GPU: NVIDIA GeForce GTX 1060M (1280 CUDA Cores, 6 GB Memory)

Jetson AGX Xavier (decentralized environment)

- CPU: ARM v8.2 64-Bit
- Memory: 16 GB
- GPU: NVIDIA Volta GPU (512 CUDA Cores with 64 Tensor Cores)

Jetson Nano (for comparison)

- CPU: ARM A57 64-Bit
- Memory: 4 GB
- GPU: NVIDIA Maxwell GPU (128 CUDA Cores)

Cameras

- FLIR Systems, Inc. Blackfly S GigE
- Frame rate: 20 fps
- Image size: 1280 (W) $\times$ 960 (H) $\times$ 3 (byte/pixel)

Hub

- NETGEAR GS108PEv3
- Number of 10/100/1000 Base-T RJ45 ports: 8 (4 PoE 802.3af Ports included)

Object detection result

- Self-defined ROS topic
- Approximately 1 KB per detected object, including bounding box, detection score, and detected object's category

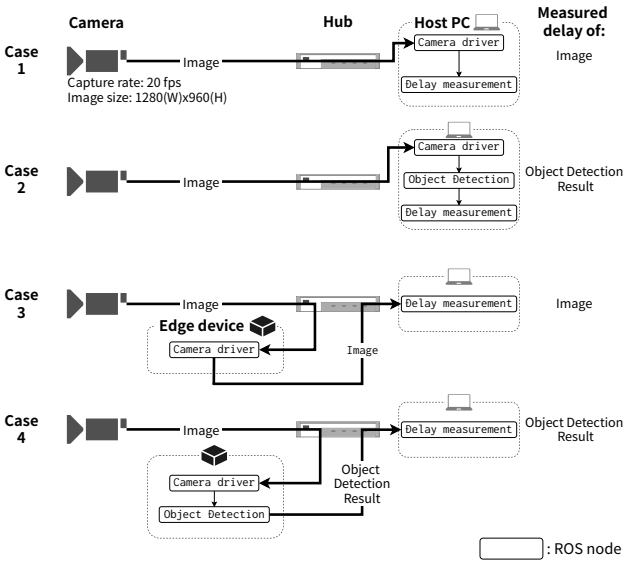

Fig. 2. Connection diagram for evaluating data delays on decentralized processing. In each case, we off-loaded camera driver and object detection modules to the edge devices to measure the effect of decentralized processing.

5.2. Delays Depending on the Presence of Decentralized Processing

Figure 2 depicts a connection diagram of the evaluations. Measured delays are as follows:

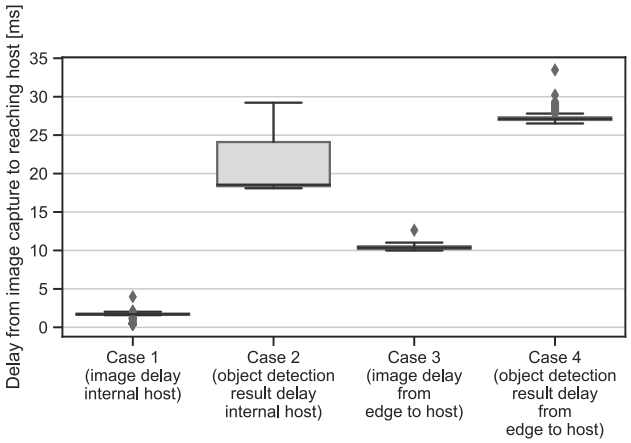

Fig. 3. Delay comparison between w/ and w/o decentralized processing.

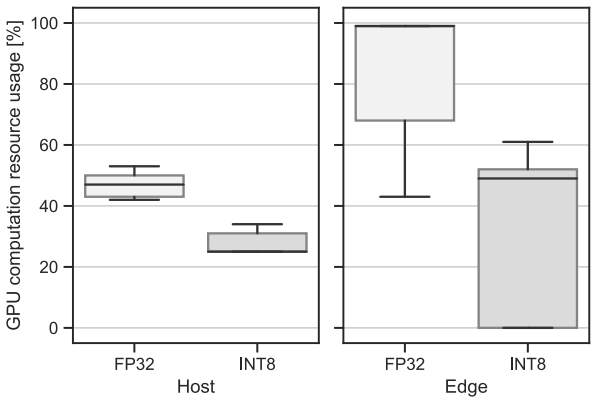

Fig. 4. GPU computational resource utilization in different operation precisions, in cases which object detections are executed on the host (left) and on the edge device (right).

- Case 1:
  Elapsed time until the host PC subscribed the images published by a camera driver node running on the same host.
- Case 2:
  Elapsed time until the host PC subscribed the object detection results published by an object detection node running on the same host. A camera driver node was also running on the same host in this case.
- Case 3:
  Elapsed time until the host PC subscribed the images published by a camera driver node running on the edge device.
- Case 4:
  Elapsed time until the host PC subscribed the object detection results published by an object detection node running on the edge device. A camera driver node was running on the edge device in this case.

In that evaluation, we set the capture rate of every camera to 20 fps. We assumed that this capture rate was enough for autonomous driving systems, because some kinds of sensor devices, such as 360° LiDAR units, work at 10 fps, and the systems must work in synchrony with these devices. To measure each delay, we took subtractions between the timestamps of target topics (i.e., the time at which the image data were fed to the system by the driver node) and the moment at which the host PC received these topics, as described in Section 4.1. Since we implemented the system through ROS, image data were not available to the other functions (i.e., the other ROS nodes) until the camera driver node fed the data to the ROS network. Hence, the delays we measured represent the elapsed time between the moment when the image data were available and the moment when the desired data reached the host. **Fig. 3** indicates the delays measured in each case of **Fig. 2**. In Case 1, in which image data were published and subscribed inside the host, the delay was approximately 1.47 ms on average. In Case 3, in which image data were published by the edge device and subscribed by the host, the delay was approximately 10.4 ms on average. This corresponds to the delay of image transfer when solely the camera driver was off-loaded to the edge device (i.e., decentralized unit). We observed that the simple off-loading of the camera driver increased the delay by approximately ten times. Since systems using decentralized processing sometimes suffer from this kind of delay, system-wide evaluation should be considered to benefit from decentralized processing. When publishing images and object detection were performed on a single host, which corresponds to Case 2, the measured average delay was approximately 21.0 ms. We can assume that this delay roughly corresponds to the elapsed time for the object detection task. However, as the exact elapsed time for the object detection task should vary depending on the GPUs, we measured Case 4 to derive the total delay when the camera driver and object detection modules were off-loaded to the edge device. When publishing images and object detection were performed on the edge device (Case 4), it took approximately 27.0 ms to reach the detection results to the host. Although the largest delay was observed in Case 4, the measurements revealed that the results of object detection could reach the host without frame drop even though it was off-loaded to the decentralized edge devices if the capture rate of the camera was 20 fps (= 50 ms/frame).

5.3. Effects of Network Quantization on Resource and Power Consumption

Figure 4 depicts the consumption of GPU computational resources on each platform when FP32 and INT8 inference engines were exploited. The left side of the figure indicates the measurement results for GPU on the host, and the right side is for GPU on the edge device. Regarding the host, although approximately 46.8% of the resources were occupied when the FP32 inference engine was in operation, it decreased to approximately 27.8% when the INT8 inference engine was in operation. Re-

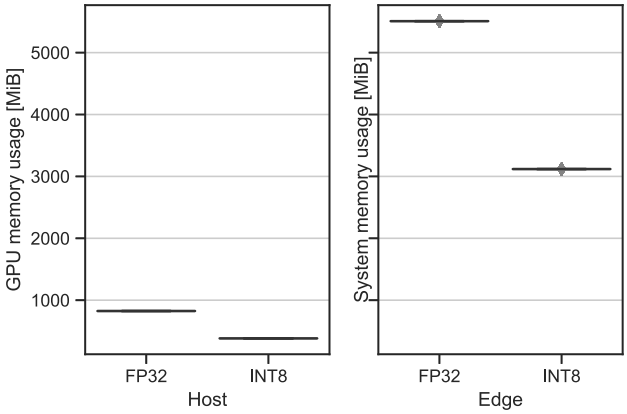

Fig. 5. GPU memory utilization in different operation precisions measured during the same experiments in **Fig. 4**.

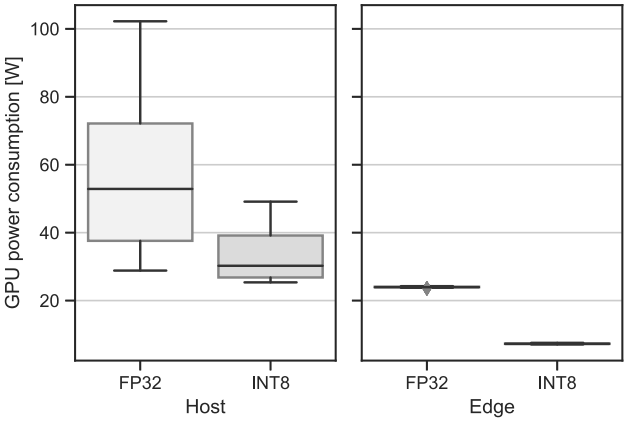

Fig. 6. Power consumption of GPU in different operation precisions measured during the same experiments in **Fig. 4**.

garding the edge device, it was approximately 86.3% and 34.4% on average for the FP32 and INT8 inference engines, respectively. As shown in **Fig. 4**, the quantized network led to the decreased consumption of the GPU computational resources during inference at both the host and the edge device. Especially at the edge device, the average of GPU computational resources consumed by the INT8 inference engine was approximately 2.51 times lower than that consumed by the FP32 inference engine. The consumption rate was different between the host and the edge device. This might partially result from the difference in the number of cores contained in each GPU; however, it suggests that TensorRT performed different adjustments for each GPU on the host and the edge device to generate inference engines.

Figure 5 indicates the memory consumption during the same experiments in **Fig. 4**. Note that the measurement for the host (left side of the figure) indicates the consumption of GPU memory, whereas the measurement for the edge device (right side of the figure) indicates the consumption of the whole system memory. This is because there is no available interface to measure solely GPU memory consumption on the edge device. Reusing GPU memory region once allocated is generally recommended to maximize performance, because the allocation of GPU memory is an expensive operation [i]. The standard deviations of memory consumption observed for both GPUs were nearly zero. A possible reason is that the memory region was reused in the engines generated by TensorRT. Similar to the computational resource, the memory consumption was degraded on both GPUs when the network was quantized. Especially on the edge device, the consumption was degraded by approximately 1.77 times on average.

The power consumption of the GPUs on the same experiments in **Fig. 4** is illustrated in **Fig. 6**. We observed the same trends of the effect of network quantization for power consumption; it decreased on both GPUs. The lowest power consumption was 7.31 W on average, which was achieved by the combination of edge processing and INT8 inference engine.

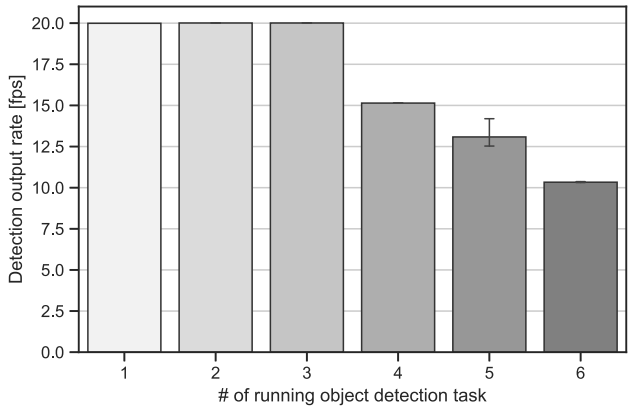

Fig. 7. Output rate of object detection results in the centralized environment.

5.4. Limitation of Centralized Processing

To discover the limitation of centralized processing, we measured the ROS topic (i.e., object detection results) rate when we assigned the number of running object detection nodes from one to six on the host. This measurement was performed with the setting of Case 2 of **Fig. 2**. To simulate a situation where a single host processed data from multiple cameras, the multiple object detection nodes subscribed images published by a single camera driver node in this measurement. **Fig. 7** indicates the measurement results. When the number of object detection nodes exceeded three, the topic rate was lower than 20 fps. Because we set cameras capture rate to 20 fps, frame dropping occurred. **Fig. 8** demonstrates the consumption of GPU computational resources in this experiment. As shown in **Fig. 7**, frame dropping is observed when running nodes are 4–6, and the average consumptions of GPU resources are over 80% of those situations. This result suggests that centralized processing could decrease the performance of perception tasks if it is in charge of multiple sensors.

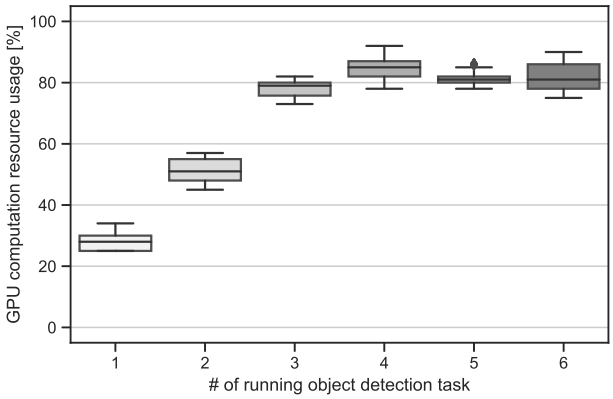

Fig. 8. Utilization of GPU computational resource on the host with different number of tasks.

Table 3. Precision and recall comparison of different operation precisions for six specific categories on MS COCO val 2014.

Metrics	IoU[6] range	Area[7]	maxDets[8]	FP32	INT8
Average precision	0.50	All	100	**0.552**	**0.498**
	0.75	All	100	0.373	0.353
	0.50 : 0.95	Small	100	0.098	0.069
	0.50 : 0.95	Medium	100	0.343	0.304
	0.50 : 0.95	Large	100	0.587	0.583
Average recall	0.50 : 0.95	All	1	0.237	0.226
	0.50 : 0.95	All	10	0.372	0.342
	0.50 : 0.95	All	100	0.375	0.344
	0.50 : 0.95	Small	100	0.114	0.077
	0.50 : 0.95	Medium	100	0.381	0.335
	0.50 : 0.95	Large	100	0.645	0.635

5.5. Effects of Network Quantization on Detection Performance Using Different Series of the Edge Device

The effects on detection accuracy and speed by changing the operation precision of the network are evaluated in this section. Additionally, we compare the execution time of object detection using different series of Jetson in the latter part of this section. Although Jetson Nano has a smaller body and lower power consumption, Jetson AGX Xavier is superior in terms of the computation power aspect. By comparing these two, we explore a more appropriate one for the edge devices of our model.

Table 3 indicates the average precisions and average recalls of object detection achieved by FP32 and INT8 inference engines. We employed MS COCO 2014 validation image set [j] to calculate average precisions and recalls. To simulate the environment perception of autonomous driving, we utilized data that belong to solely six categories (`car`, `pedestrian`, `bus`, `truck`, `bicycle`, `motorcycle`) contained in the dataset to calculate these metrics. The value of each item was slightly lowered after changing the operation precision from FP32 to INT8. SSD512 [7], which is another state-of-the-art detector, achieved mAP-50 $= 0.485$ on MS COCO 2015 test image set. Although the direct comparison is not fair because the used dataset was different, the INT8 inference engine achieved a detection accuracy comparable to the other state-of-the-art detector.

Figure 9 illustrates the inference times of object detection achieved by each power mode preset using inference engines with different operation precisions. With the help of Tensor cores and deep learning accelerators (DLAs), which can process FP16 and INT8 operation fast, the execution times for every power mode preset were significantly improved by changing the operation precision. Furthermore, the execution time difference between power mode presets was also significant. Therefore, the difference of the maximum frequencies of GPUs and DLAs between each power mode preset mainly caused the performance difference. The power budget "n/a" with the INT8 inference engine achieved approximately 24.5 ms ($\approx$ 40.8 fps) for average inference time. For compar-

6. IoU: Intersection over union.
7. According to https://cocodataset.org//#detection-eval [Accessed January 24, 2021], the definition of object area sizes are as follows:
   small: area $< 32^2$ pixel
   medium: $32^2 <$ area $< 96^2$ pixel
   large: 96^2 pixel $<$ area.
8. maxDets: thresholds on max detections per image.

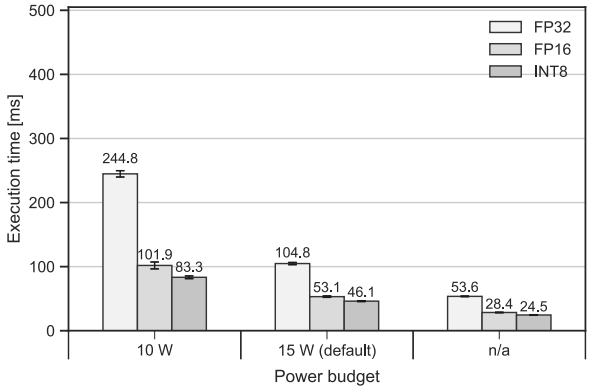

Fig. 9. Execution time comparison regarding power budget and operation precision on Jetson AGX Xavier.

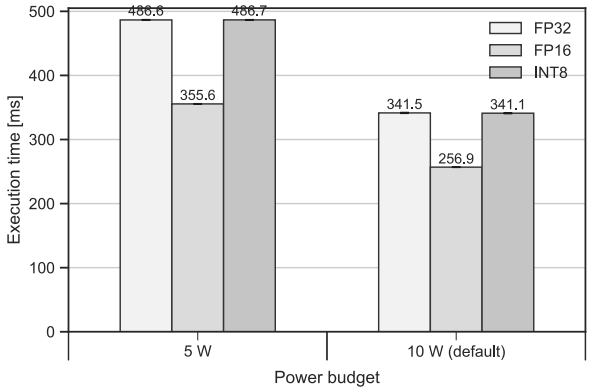

Fig. 10. Execution time comparison regarding power budget and operation precision on Jetson Nano.

ison, we conducted similar measurements using Jetson Nano [d]. The results are demonstrated in **Fig. 10**. Two power mode presets are provided for Jetson Nano, and the power budgets for the presets are 5 W and 10 W, respectively (10 W is the factory default). Because Jetson Nano contains CUDA cores that are four times fewer than Jetson AGX Xavier, the overall trend of processing speed was slower compared to Jetson AGX Xavier. Moreover, changing the operation precision did not improve the inference speed significantly. This is because the generation of GPUs equipped on Jetson Nano is Maxwell, which does not contain any Tensor cores and DLAs. From the point of autonomous driving view, approximately 260 ms execution time ($\approx$ 3.8 fps), which is the highest processing speed achieved by Jetson Nano, is hard to accept as onboard processing delay. Through this comparison, we concluded that Jetson AGX Xavier was preferable to Nano until that point.

5.6. Discussion

Here, we discuss the validity of the decentralized processing for environment perception tasks for autonomous driving.

Table 4. Comparison of network traffic amount w/ and w/o decentralized processing under our experimental setting.

Processing type	Breakdown				Approx. total [MB/s]
w/o decentralized (i.e., centralized)	Height [pixel]	Width [pixel]	byte/ pixel	fps	70
	960	1280	3	20	
w/ decentralized	data-size/ object [KB]		objects/ frame	fps	$0.02N$
	1		N	20	

As shown in the experimental setup, we set the image size to 1280 (W) $\times$ 960 (H) and frame rate to 20 fps as the camera configuration. We also deployed a self-defined ROS topic to transfer the object detection results, which was approximately 1 KB per object. Simplified thinking, the amount of data fed to the system by one camera was 1280 (W) $\times$ 960 (H) $\times$ 3 byte/pixel $\times$ 20 fps $= 73728000 \approx 70$ MB/s under this configuration. In contrast, provided that solely object detection results were transferred, the amount of transferred data decreased in orders of magnitude and was $1 \times N$ [objects/frame] $\times$ 20 fps $\approx 20N$ [KB/s]. The comparison of the amount of data transferred is summarized in **Table 4**. Considering the network load of the host machine, decentralized processing was more tolerant against system scaling because this data amount increased linearly according to the number of external sensors.

To recognize the surrounding circumstances of the ego-vehicle, we assumed a system using multiple cameras that equipped a standard field of view (FOV) lens and detected objects by processing captured images. Typically, a standard FOV camera lens has a 25°–50° view angle. In such a case, at least six to seven cameras are required to capture all directions. Wide FOV cameras, including fisheye cameras, may be the alternatives to capture surrounding circumstances using fewer cameras. In [12], the authors published a dataset for autonomous driving that includes images captured by fisheye cameras. They reported that they conducted a baseline experiment of object detection, which used Faster R-CNN [13] on fisheye-camera images, and the detection accuracy was significantly lower than that achieved in other datasets. They expect that the accurate object detection on fisheye camera images is difficult owing to large lens distortion ("*the orientation of objects in the periphery of images being very different from central region*"). For the perception systems of autonomous driving, one-stage detectors, including YOLOv3, are preferable to two-stage detectors, including Faster R-CNN, because fast and online object detection is required. On the other hand, one-stage detectors typically suffer from the lower accuracy of bound-

ing box regression compared to two-stage detectors [14]. Considering these facts, it is suggested that the detection accuracy of one-stage detectors may degrade if the detectors receive wide FOV images with large distortion as input. For this reason, processing images captured by multiple cameras with small distortion is promising to detect objects on surrounding circumstances accurately.

If a single machine is in charge of all those cameras, there are concerns for the frame dropping because of the heavy burden of transferring data and computation. If we assume a centralized system consisting of a single host machine that is in charge of six cameras set to the same configuration of our experimental setup, the transferred image data theoretically occupy the network of the system 420 MB/s constantly. Such a system will suffer from frame dropping, as shown in **Fig. 7**, as well as from the high occupancy rate of the network. On the other hand, if we introduce decentralized units to each camera, the network burden decreases to $0.12N$ [MB/s]. Here, N is the number of detected objects per frame and is typically under 100, even for images captured while driving in a complicated urban area. Even if we put an additional assumption of $N = 100$ objects/frame, the network burden is 12 MB/s, which is 35 times lower than that of the centralized system. Regarding the host machine, it involves little computational burden for object detection as the burden is off-loaded to decentralized units, and other tasks can safely utilize the resources of the host. When it comes to autonomous driving, the frame dropping and the resource occupation by single task prevent the proper operation of the whole system and lead to serious accidents. Therefore, using multiple cameras to capture surrounding circumstances and processing images not on a single machine but on decentralized computational resources might be the proper practice; moreover, it could be a realistic solution for the actual deployment of autonomous driving systems.

Increasing power consumption by decentralizing computational resources remains a concern. As shown in **Fig. 6**, the power consumption of the GPU is decreased by embedded GPU and network quantization. It should be noted that the presented values indicate the power consumption of GPU exclusively and do not include large power consumers, such as DRAM memory. As described in Section 4.3, we did not set any limitations on the power budget of the edge device on the experiments of Section 5.2. The power consumption of the whole system is assumed to be decreased when setting limitations to the power budget. For example, as shown in **Fig. 9**, the average inference time was approximately 46.1 ms (≈ 21.7 fps) in the case of the combination of 15 W power budget (factory default for Jetson AGX Xavier) and INT8 inference engine. Thus, we assume that the inference process itself is possible to run over 20 fps even with the limitation of the power budget. Furthermore, as shown in **Figs. 4**–**6**, using the INT8 inference engine decreased the consumption of computational resources, memory, and power; moreover, the running was approximately 40 fps at best. Besides setting limitations of power budget, processing a few cameras using a single edge device might also suppress the total power consumption per camera.

Increasing system complexity is another concern for decentralized processing. In general, introducing decentralized processing makes the system more complex; it sometimes brings the system reliability (i.e., the ability of the system to function without failure) to be decreased. The software complexity of the proposed system did not increase drastically when we introduce decentralized processing because of the ROS functionality, however, hardware complexity increased indeed. This increase in the hardware complexity caused by introducing decentralized processing is inevitable. On the other hand, the system's robustness against failure should also be considered. At the high-level (four and above) autonomous driving systems, system failure during driving operations must be treated by the system itself properly (e.g., stop the car at the safe place) since such systems do not suppose driving tasks' fallback on the drivers [1]. On centralized processing systems, margins of the computational resources to treat the failure are estimated to be limited because the single computation unit should address all computation processing. In contrast, the resource margins are considered to be relatively enough if we employ a decentralized processing fashion because the fashion can avoid concentrating computation burden to a specific location of the systems. From this point of view, we consider that decentralized systems have higher safety against failure than centralized ones.

As a limitation on the application of this study, we did not assume high velocity driving such as driving on highways with no velocity limit. Since we suppose enough velocity to drive in an urban area, such as 60 km/h, we set the camera frame rate to 20 fps as the experimental setup; this configuration is not enough to drive at high velocity. More detail, including camera frame rate, network protocol, and ROS topic (i.e., transferred data) format, should be reconsidered from the perspective of processing speed to relax this limitation. However, we believe the advantage of decentralized processing, including decreasing power consumption and avoiding concentration of computation burden, remains to construct autonomous driving systems.

6. Conclusions

In this paper, we explored the validity of decentralized processing for the perception tasks of autonomous driving systems to decrease the computational and network load of a host computer. To clarify the effect of introducing decentralized processing, including the delay caused by data transfer and the throughput of the whole system, we constructed a prototype of a system that performed an object detection task on embedded oriented GPUs and evaluated it.

The results of our quantitative evaluations revealed that it was approximately 27 ms on average from the moment at which an image was fed to the system to the moment at

which the object detection results of the image reached the host. This indicates that, although the measured delay included the overhead caused by decentralized processing, no frame dropping occurred in the detection task when the capture rate of cameras was around 20 fps. In addition to the computational off-loading, our experimental evaluations demonstrate that the network load of the host was also decreased in orders of magnitude. We also confirmed that the embedded oriented GPUs and the quantization of inference networks were effective to implement fast object detection. Their combination achieved, at best, 40.8 fps for image-based object detection as well as low consumption of computational resources, memory, and power. Furthermore, the evaluation results suggest that the degradation of detection accuracy caused by the quantization of the inference network was insignificant, and the accuracy was comparable with other state-of-the-art object detectors. From these facts, we conclude that autonomous driving systems could benefit from decentralized processing, even after considering the delay caused by its introduction.

To apply decentralized processing to onboard systems, there is still room for further consideration regarding power consumption. We proved that the network quantization decreased the power consumption by 3.3 times on embedded GPUs, and the object detection inference ran with approximately 7.3 W per camera on average. However, since the result indicated the power consumption of GPU exclusively, the whole system should consume more power because of large power consumers, such as memory devices on decentralized units. As countermeasures for this problem, changing the power mode (i.e., setting limitation of power consumption) of decentralized units or processing several cameras using one decentralized unit could be promising. Future work needs to design the decentralized system in more detail. Considering other factors, including camera fps and required processing speed for object detection, the balance of the number of decentralized units and total power consumption should be designed for the actual deployment of the system.

Acknowledgements

This paper is based on results obtained from a project, JPNP20016, subsidized by the New Energy and Industrial Technology Development Organization (NEDO).

References:

[1] SAE On-Road Automated Driving Committee and others, "SAE J3016. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," Technical Report, SAE Int., 2016.

[2] JSAE Standardization Board, "JASO TP 18004. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," Technical Report, Society of Automotive Engineers of Japan, Inc., 2018.

[3] S. Mittal, "A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform," J. of Systems Architecture, 2019.

[4] M. Eisenbach, R. Stricker, D. Seichter, A. Vorndran, T. Wengefeld, and H.-M. Gross, "Speeding up deep neural networks on the jetson tx1," Proc. of the Workshop on Computational Aspects of Pattern Recognition and Computer Vision with Neural Systems at Int. Joint Conf. on Neurlal Networks, 11, 2017.

[5] N. Tijtgat, W. Van Ranst, T. Goedeme, B. Volckaert, and F. De Turck, "Embedded real-time object detection for a uav warning system," Proc. of the IEEE Int. Conf. on Computer Vision Workshops, pp. 2110-2118, 2017.

[6] R. Giubilato, S. Chiodini, M. Pertile, and S. Debei, "An evaluation of ROS-compatible stereo visual SLAM methods on a nVidia Jetson TX2," Measurement, Vol.140, pp. 161-170, 2019.

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," Proc. of the European Conf. on Computer Vision, 2016.

[8] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv: 1804.02767, 2018.

[9] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," Proc. of Workshop on Open Source Software of the IEEE Int. Conf. on Robotics and Automation, Vol.3, p. 5, 2009.

[10] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and H. Tsuyoshi, "An Open Approach to Autonomous Vehicles," IEEE Micro, Vol.35, No.6, pp. 60-68, 2015.

[11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," Proc. of European Conf. on Computer Vision, pp. 740-755, 2014.

[12] S. Yogamani et al., "Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving," Proc. of the IEEE/CVF Int. Conf. on Computer Vision, pp. 9308-9318, 2019.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Advances in Neural Information Processing Systems, pp. 91-99, 2015.

[14] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A Survey of Deep Learning-Based Object Detection," IEEE Access, Vol.7, pp. 128837-128868, 2019.

Supporting Online Materials:

[a] NVIDIA, Jetson TX1. https://developer.nvidia.com/embedded/jetson-tx1 [Accessed January 24, 2021]

[b] NVIDIA, Jetson TX2. https://developer.nvidia.com/embedded/jetson-tx2 [Accessed January 24, 2021]

[c] NVIDIA, Jetson AGX Xavier. https://developer.nvidia.com/embedded/jetson-agx-xavier [Accessed January 24, 2021]

[d] NVIDIA, Jetson Nano. https://developer.nvidia.com/embedded/jetson-nano [Accessed January 24, 2021]

[e] arm, Mali series. https://www.arm.com/en/products/silicon-ip-multimedia [Accessed January 24, 2021]

[f] NVIDIA, TensorRT. https://developer.nvidia.com/tensorrt [Accessed January 24, 2021]

[g] The Autoware Foundation, "Autoware." https://gitlab.com/autowarefoundation/autoware.ai [Accessed January 24, 2021]

[h] COCO Consortium, "COCO 2017 Val images." http://images.cocodataset.org/zips/val2017.zip [Accessed January 24, 2021]

[i] NVIDIA, "CUDA C++ Best Practices Guide," 2019. https://docs.nvidia.com/cuda/pdf/CUDA_C_Best_Practices_Guide.pdf [Accessed January 24, 2021]

[j] COCO Consortium, "COCO 2014 Val images." http://images.cocodataset.org/zips/val2014.zip [Accessed January 24, 2021]

Name:
Manato Hirabayashi

Affiliation:
Graduate School of Information Science, Nagoya University

Address:
Furo-cho, Chikusa-ku, Nagoya, Aichi 464-8603, Japan

Brief Biographical History:
2013 Received B.E. degree from Nagoya University
2015 Received M.S. degree from Nagoya University
2015- Ph.D. Student, School of Information Science, Nagoya University

Name:
Yukihiro Saito

Affiliation:
Tier IV, Inc.

Address:
Jacom Building, 1-12-10 Kitashinagawa, Shinagawa-ku, Tokyo 140-0001, Japan

Brief Biographical History:
2015 Received B.S. in Information Science from Ritsumeikan University
2017 Received M.S. in Information Science from Ritsumeikan University
2018- Tier IV, Inc.

Name:
Kosuke Murakami

Affiliation:
Tier IV, Inc.

Address:
Jacom Building, 1-12-10 Kitashinagawa, Shinagawa-ku, Tokyo 140-0001, Japan

Brief Biographical History:
2017- Tier IV, Inc.

Name:
Akihito Ohsato

Affiliation:
Tier IV, Inc.

Address:
Jacom Building, 1-12-10 Kitashinagawa, Shinagawa-ku, Tokyo 140-0001, Japan

Brief Biographical History:
2014 Received B.S. degree from Tokyo University of Science
2016 Received M.S. degree from Tokyo University of Science
2016- Sony Corporation
2017- Tier IV, Inc.

Name:
Shinpei Kato

Affiliation:
Graduate School of Information Science and Technology, The University of Tokyo

Address:
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan

Brief Biographical History:
2008 Received Ph.D. degree from Keio University
2009- The University of Tokyo / Carnegie Mellon University
2012- Nagoya University
2016- The University of Tokyo

Name:
Masato Edahiro

Affiliation:
Graduate School of Informatics, Nagoya University

Address:
Furo-cho, Chikusa-ku, Nagoya, Aichi 464-8603, Japan

Brief Biographical History:
1985- NEC Corporation
1999 Received Ph.D. degree from Princeton University
2011- Nagoya University