IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON DATA MINING FOR INTERNET OF THINGS

Received November 5, 2019, accepted November 15, 2019, date of publication November 25, 2019,
date of current version December 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2955761

Practical Monitoring of Undergrown Pigs for IoT-Based Large-Scale Smart Farm

SUNGJU LEE[1], HANSE AHN[1], JIHYUN SEO[1], YONGWHA CHUNG[1], DAIHEE PARK[1], AND SUNGBUM PAN[2], (Member, IEEE)

[1]Department of Computer Convergence Software, Korea University—Sejong, Sejong 30019, South Korea
[2]Department of Electronics Engineering, Chosun University, Gwangju 61452, South Korea

Corresponding authors: Yongwha Chung (ychungy@korea.ac.kr) and Daihee Park (dhpark@korea.ac.kr)

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education (2018R1D1A1A09081924 and 2019R1H1A1080140, and NRF-2017R1A6A1A03015496).

ABSTRACT Taking care of individual pigs is important in the management of a group-housed pig farm. However, this is nearly impossible in a large-scale pig farm owing to the shortage of farm workers. Therefore, we propose an automatic monitoring method in this study to solve the management problem of a large-scale pig farm. Particularly, we aim to detect undergrown pigs in group-housed pig rooms by using deep-learning-based computer vision techniques. Because the typical deep learning techniques require a large computational overhead (*i.e.*, Mask-R-CNN), fast and accurate detection of undergrown pigs on an IoT-based embedded device is very challenging. We first obtain the video monitoring data of group-housed pigs by using a top-view camera that is installed in the pig room, and then detect each moving pig by combining image processing and deep learning techniques. Gaussian Mixture Model is used to detect moving frames and moving objects. In embedded device implementations, by applying deep learning (*i.e.*, TinyYOLO3) to a few frames only with a large number of pixel changes, embedded GPUs can be used efficiently, satisfying the real-time requirement. As a subsequent step, we check the acceptable conditions of the posture and separability from each video frame of the continuous video stream. Finally, to compute the relative size of each pig quickly and accurately, we develop image processing steps to complement the result of deep learning with minimum computational overhead. Furthermore, by pipelining the CPU and GPU steps of a continuous video stream, we can hide the additional image processing time. Based on the experimental results obtained from an embedded device, we confirm that the proposed method can automatically detect undergrown pigs in real-time, by working as an early warning system without any human inspection or measurement of actual weight by a farm worker.

INDEX TERMS Smart farm, pig monitoring, computer vision, image processing, deep learning.

I. INTRODUCTION

IoT techniques have been used widely for several monitoring applications. For example, IoT-based monitoring techniques have been applied to pig farms to control the temperature and humidity automatically [1], [2]. As a subsequent step, the Korean government has planned to upgrade this first-generation smart pig farm by providing additional intelligence with a camera and deep learning techniques. The goal of this upgrade is to take care of every pig with the help of intelligent monitoring techniques. During our survey, we visited a large-scale pig farm in Korea. This farm consisted of more than 20,000 pigs (in approximately 1,000 pig rooms) and 10 farm workers. Because the work at a pig farm is labor-intensive, it is very difficult to hire the required number of farm workers in Korea. Therefore, each farm worker in a Korean pig farm is so highly occupied that taking care of every pig is nearly impossible.

According to statistics published by the Korean government, approximately 5 million out of 20 million pigs die every ''typical'' year. In addition, in a ''special'' year, additional pigs die because of a specific disease. For example, approximately 3 million pigs died from the foot-and-mouth disease (FMD) in 2010. In Korea, typical reasons such as the wasting disease and aggressive behavior caused by the closed environment of group-housed pig rooms are the main reasons

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Wei Tsai.

for pig death, when compared with special reasons such as FMD. However, unlike with FMD, if these cases are detected early, we can save the sick pigs.

Early detection of diseases in pigs is possible by monitoring individual pigs. However, as each farm worker is highly occupied, human inspection of 2,000 pigs (the average number of pigs per farm worker was approximately 2,000 for the five Korean pig farms we had visited) is nearly impossible. The goal of the second-generation smart pig farm is to monitor every pig continuously without any human inspection. As we have already reported the monitoring methods for the wasting disease [3], aggressive behavior [4], and touching-pigs [5] previously, this study focuses on detecting undergrown pigs. In several cases, the undergrown pigs die if the necessary care is not provided at the right time.

Detecting undergrowth in pigs by using computer vision techniques requires highly accurate detection of each pig in a group-housed pig room (known as *instance segmentation* used to detect each object in pixel-level accuracy). The recent developments in deep learning techniques can be applied to several computer vision tasks, including a complicated task such as instance segmentation. Thus, detecting undergrown pigs by using deep-learning-based computer vision techniques becomes possible. However, we need to provide an automatic monitoring method that is cost-effective for large-scale pig farms.

To provide a practical method, we should consider an embedded device and the concept of ''on-device'' computing [6]–[9]. However, an embedded device has a limited computing power than the typical PCs, and the instance segmentation requires a large computational overhead. This conflicting situation is not a special case, but a general case in applying complicated deep learning techniques to IoT-based applications. To solve this conflicting situation, we first select a very fast deep learning technique for *object detection* (*i.e.*, detecting each object in bounding box-level accuracy), which is a less complicated computer vision task than the instance segmentation, and thus, it can be executed on an embedded device. Then, to compute the size of each pig quickly and accurately, we develop image processing steps to complement the result of the very fast deep learning, with minimum computational overhead. Ultimately, we compare the relative size of each pig in a closed pig room to detect the undergrown pigs and send alarms to the farm worker. Figure 1 shows an overview of the proposed monitoring system.

This paper is organized as follows: Section 2 summarizes the previous methods; Section 3 describes the proposed method that was used to detect the undergrown pigs efficiently; Section 4 explains the details of the experimental results; and Section 5 presents the conclusion.

II. RELATED WORKS

Several studies on monitoring pigs automatically have recently been reported [1]–[5]. In a group-housed pig room, there are several pigs, and there is a high possibility of them

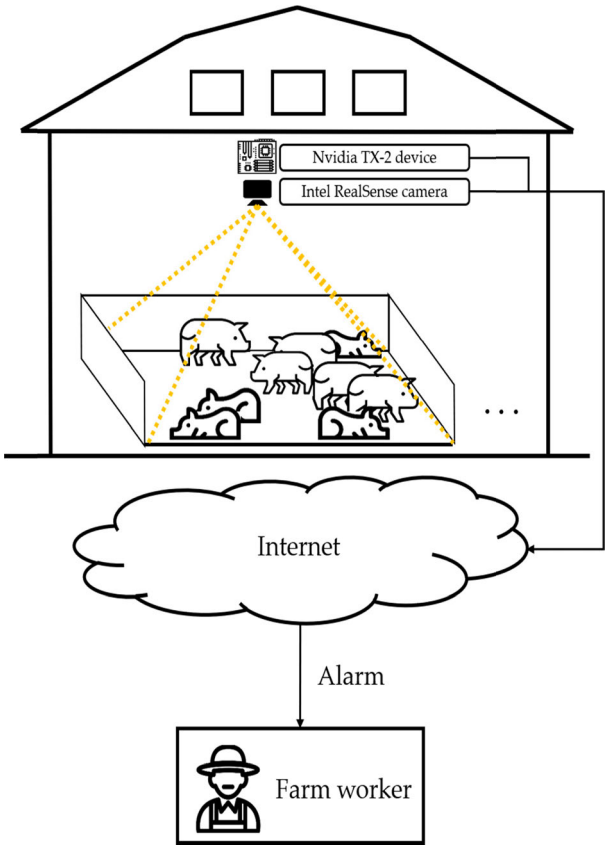
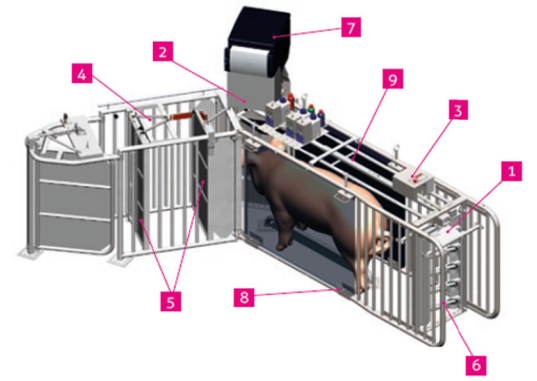


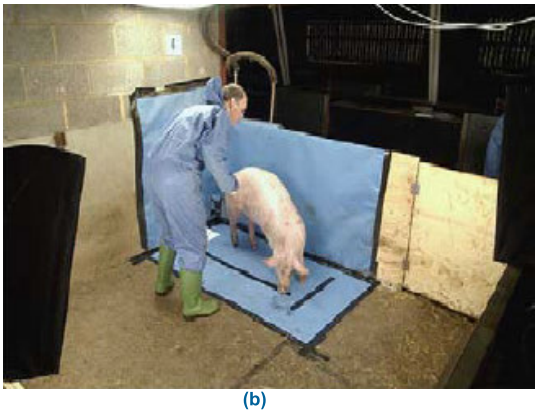
FIGURE 1. Overview of the proposed monitoring system.

touching each other; these touching-pigs need to be separated to achieve the final goal of tracking individual pigs for 24 h. Ju *et al.* reported a method for separating two touching-pigs to analyze individual pig's behavior without ID switch. However, in addition to this behavior monitoring, we need to monitor the pig's size automatically to detect undergrown pigs in a large-scale farm.

Continuous weight control of each pig is important in pig farms. Because pigs are essentially non-cooperative, measuring the actual sizes or weights of pigs manually is highly labor-intensive. For example, the manual measurement of weights of heavy pigs consumed approximately 3–5 min per pig with two farm workers [10]. This procedure stresses both farm workers and pigs. Therefore, an effective method that can measure or compute the pig's size/weight on a regular basis is necessary. The first method is to use RFID techniques to measure the actual weight of each pig, with the aid of a special facility (see Figure 2 (a)). In this method, a facility that allows only a single pig to enter at a time is provided in front of the feeding area, and the weight of the pig is measured during feeding, along with the RFID information [11]. However, the cost of this facility is relatively high (>$10,000 per pig room). Furthermore, because of the shortage of the farm workers in Korea, only the sows are being managed with RFID in several pig farms. In fact, this situation is same in other countries. Therefore, researchers have been studying the computer vision techniques to compute the size



(a)



(b)

(c)

FIGURE 2. Previous methods used for pig's weight measurement (direct) or image-based computation (indirect). (a) Measurement of a single pig with a high-cost facility [11]. (b) Image-based computation of a single pig with a special facility and an aid from human [13]. (c) Image-based computation of group-housed pigs with an aid from human [25].

of each pig indirectly (rather than the actual weight of each pig) since 1990 [12], and some commercial systems such as Weight-Detect, eYeScan, and Pigwei have been released recently.

Most of the previous methods and commercial systems computed the size of a single pig by assuming a single pig in a specially organized measurement area [13]–[23]. However, this assumption either requires a large number of labor hours or a special facility to obtain an image of a single pig at a time (see Figure 2 (b)). Therefore, our goal is to compute the sizes of group-housed pigs in a cost-effective way without any human intervention or a special facility.

Only a few studies have computed the size of each pig in a group-housed pig room without a special facility [24]–[27]. These methods do not incur any managing overhead costs after a top-view camera is installed. However, they use different "markers" (done by humans) on the back of each pig to identify them (see Figure 2 (c)). Although this marking assumption is feasible for small experiments, it is infeasible for commercial farm environments. As pigs grow rapidly, the farm workers need to maintain these markers regularly; this stresses both farm workers and pigs. Therefore, we need a "markerless" solution to compute the size of each pig in a group-housed room, without any human intervention. Furthermore, all the previous methods (except the ones that are not specified) are PC-based solutions, which may not be viable for farm owners because of their costs. We have summarized some of the previous methods that were used to compute the size of a pig in Table 1. The main contributions of this study are as follows:

- Because the proposed method uses computer vision techniques, it does not incur any managing overhead costs after a top-view camera is installed. Furthermore, the proposed method is a "markerless" solution and does not require any human intervention. To the best of our knowledge, this is the first study to detect undergrown pigs without any human intervention, and it operates by comparing the relative size of each pig in a group-housed room.
- The proposed method can be executed on an embedded device, such as NVIDIA TX-2 [28]. Although many PC-based methods have been proposed to compute the size of each pig, an embedded device-based method is proposed for the first time, to the best of our knowledge. Thus, the total cost of the proposed method (including a camera) is less than $1,000 per pig room, and therefore, this method can be applied to large-scale pig farms. Because the embedded devices have a limited computing power than the typical PCs, we combine the image processing and deep learning techniques carefully to satisfy the requirements of both accuracy and execution time.

III. PROPOSED METHOD

To provide a practical method, we should consider an embedded device and the concept of "on-device" computing [6]–[9]. That is, the data should be collected and analyzed in embedded device systems such as edge computing. Edge computing supports an efficient data processing in high-volume data processing systems, and thus, it can reduce the installation and management costs of smart farm systems, which in turn reduces the network bandwidth and analysis workload of the server. For example, an edge device

TABLE 1. Some of the previous methods that compute the size of a pig (published during 2010–2019).

Camera Type	Technique	Pigs in a Group-Housed Room	Requirement of Human Intervention	Target Platform	Real-Time Execution	Reference
2D (Area)	Image Processing	No	Yes	PC	Not Specified	[14]
2D (Area)	Learning	No	Yes	Not Specified	Not Specified	[15]
3D (Volume)	Image Processing	No	Yes	PC	Not Specified	[16]
2D (Area)	Learning	No	Yes	PC	No	[17]
2D (Area)	Image Processing	No	Yes	Not Specified	Not Specified	[18]
3D (Volume)	Image Processing	No	Yes	Not Specified	Not Specified	[19]
3D (Volume)	Image Processing	No	Yes	PC	Not Specified	[20]
3D (Volume)	Image Processing	No	Yes	Not Specified	No	[21]
2D (Area)	Deep Learning	No	Yes	Not Specified	Not Specified	[22]
3D (Volume)	Image Processing	No	Yes	Not Specified	No	[23]
2D (Area)	Image Processing	Yes	Yes (Marking)	Not Specified	Not Specified	[24]
2D (Area)	Image Processing	Yes	Yes (Marking)	Not Specified	Not Specified	[25]
2D (Area)	Deep Learning	Yes	Yes (Marking)	PC	Not Specified	[26]
2D (Area)	Image Processing	Yes	Yes (Marking)	PC	Not Specified	[27]
2D (Area)	Image Processing + Deep Learning	**Yes**	**No**	**Embedded Device**	**Yes**	Proposed Method

requires only a small size of text data (*i.e.*, relative pig's size), whereas a server requires a large size of image data (*i.e.*, video stream). In this study, we propose an embedded device-based method to detect undergrown pigs in real-time using the image processing and deep learning techniques to realize IoT-based smart pig farms. In other words, we propose a hybrid method with a deep learning technique (*i.e.*, TinyYOLO3) to detect individual pigs, and an image processing technique (*i.e.*, GMM, binarization with Othsu, and connected component) to compute the pig size, instead of the end-to-end deep learning based-method for instance segmentation (*i.e.*, Mask-R-CNN).

Because group-housed pigs can be located at the same position in a room (*i.e.*, one on top of another), the estimation of the actual weight of each pig may not be accurate in such complicated cases. Therefore, the goal of this study is to develop an automatic alarm system (without any human intervention or a high-cost facility) by analyzing the relative size and not the actual weight of each pig in a group-housed pig room. Furthermore, the technique proposed to detect the abnormality in growth rate should be considered as an aid (*i.e.*, sending alarms) rather than as a replacement for farm workers.

To accurately compute the relative size of each pig in a group-housed room, the complicated cases (*i.e.*, touching/overlapping pigs) should be considered. Therefore, we first check the acceptable conditions of the posture (*i.e.*, walking pigs) and separability (*i.e.*, separated pigs and not touching/overlapping pigs) from each video frame of the continuous video stream. Then, if the check passes, the relative size of the pig satisfying the acceptable conditions is analyzed. In addition, a solution for the resource-constrained IoT devices such as edge computing should be considered. Thus, by considering both accuracy and speed, we compute the size of each pig that satisfies the acceptable conditions by using the combination of image processing and deep learning techniques. Especially, to make the best use of the resources (*i.e.*, a multi-core CPU and a many-core GPU) of the embedded device, the heterogeneous computing is exploited by pipelining the image processing and deep learning steps.

The proposed method consists of two image processing modules (denoted as *pre-* and *post-processing*) and a deep learning processing module (denoted as *mid-processing*) as shown in Figure 3. Note that, in this study, we use the infrared information to compute the size of each ''markerless'' pig, which is used to detect undergrown pigs. As shown in Figure 4, the quality of the depth information obtained from Intel RealSense camera is not adequate to compute the size of each pig. Note that, in the depth image of pig room 2, the pigs are indistinguishable by sight.

In the pre-processing module, we first extract an infrared frame from a video stream obtained from an infrared sensor

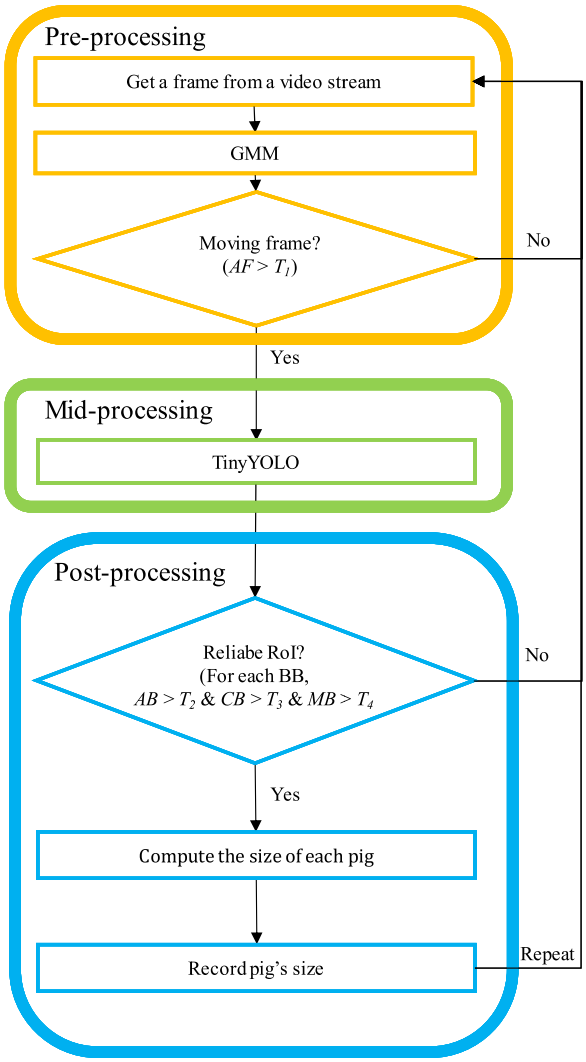


FIGURE 3. Three modules of the proposed method.

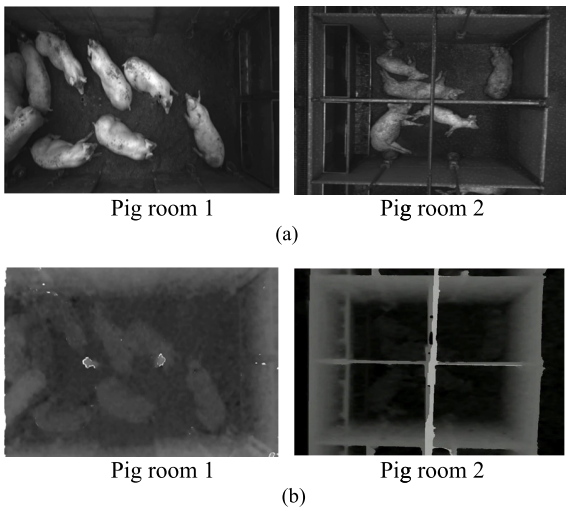


FIGURE 4. Infrared and depth images obtained from Intel RealSense camera. (a) Infrared image and (b) depth image.

(*i.e.*, Intel RealSense camera). Then, the Gaussian Mixture Model (GMM) [29] is applied to detect the moving frame for 24 h. In the mid-processing module, the individual pigs are detected using TinyYOLO [30]. Note that, TinyYOLO is a deep learning-based object detector, which generates a bounding box (BB) for each object. Especially, TinyYOLO is applied only to the video frames that include moving pigs (denoted as *moving frames*), so that it can be executed efficiently on an embedded GPU. In the post-processing module, Region of Interest (RoI) is selected by considering the acceptable posture (*i.e.*, walking pigs) and separability (*i.e.*, separated pigs). To select a moving frame and a reliable RoI for computing the pig's size, we use the amount of pixel changes of a BB (denoted as *AF* and *AB,* respectively), the confidence score of each BB (denoted as *CB*), and the minimum distance (denoted as *MB*) between adjacent BBs. Finally, the relative size of each pig is carefully computed only for the pigs that satisfy acceptable conditions, and thus, the smart farm monitoring system for detecting undergrown pigs can be realized by using an embedded device.

We summarize the definitions of *AF*, *AB*, *CB*, and *MB* in Equation (1), (2), (3), and (4), respectively:

$$AF = \frac{\text{Amount of pixel changes}}{\text{Amount of total pixels in a frame}}, \tag{1}$$

$$AB = \frac{\text{Amount of pixel changes}}{\text{Amount of total pixels in a BB}}, \tag{2}$$

$$CB = \text{Confidence score of a BB}, \tag{3}$$

$$MB = \frac{\text{Minimum distance of each BB}}{\text{Length of frame width or height}}, \tag{4}$$

where *AF* is used to reduce the total execution time, and *AB*, *CB*, and *MB* are used to detect the reliable RoI for the walking and separated pigs.

A. DECISION OF MOVING FRAMES

GMM is widely used for background modeling where the background is subtracted using each pixel modeled as mixed Gaussian distributions [29]. In this study, GMM is used to detect the moving frame to reduce the total execution time. That is, TinyYOLO is executed only when *AF* is larger than the threshold value (denoted as T_1), and thus, we can reduce the overall execution time. Figure 5 (b) illustrates a moving frame detected by using GMM. In addition, GMM is used to select reliable RoIs by considering the acceptable posture. However, when several adjacent pigs are moving at the same time, it is difficult to separate each pig accurately (see Figure 5 (c)). Therefore, we need a way to detect separated pigs to compute the size of each pig accurately, even when two or more pigs are moving close to each other at the same time.

B. DECISION OF RELIABLE ROI

To detect the individual pigs, we use TinyYOLO based on deep learning. YOLO [31] is a CNN-based single-shot object detector whose execution time is much faster than those of many-shot object detectors such as R-CNN [32], Fast R-CNN [33], and Faster R-CNN [34]. Because of its fast

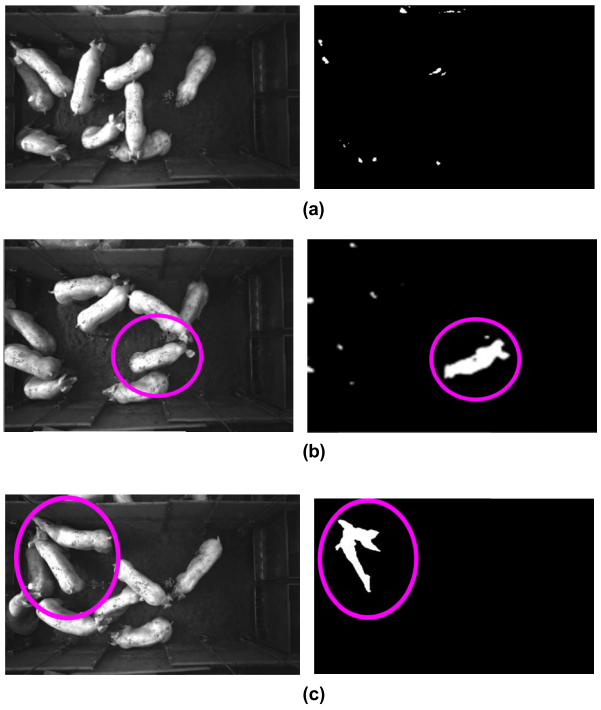


FIGURE 5. Illustration of moving frames obtained by using GMM. (a) Frame with low *AF* (*i.e.,* not moving frame). (b) Frame with high *AF* (*i.e.,* moving frame). (c) Frame with high *AF* (when two adjacent pigs are moving).

execution time, YOLO is one of the most widely used object detectors. Furthermore, TinyYOLO [30] is a light-weight version of YOLO (*i.e.*, TinyYOLO has a fewer convolutional layers than YOLO) that is used for embedded device implementations. The details of TinyYOLO can be found in [30].

Although the pigs can be detected by TinyYOLO, it is difficult to compute the size of each pig accurately because of the inaccuracy in the size of each BB caused by the pig's posture and direction. Figure 6 displays the difference between the BB and pixel sizes for various postures of the same pig. In Figure 6, the sizes of BBs are 38,413 (*i.e.*, posture 1 of vertically walking), 63,226 (*i.e.*, posture 2 of diagonally walking), and 34,882 (*i.e.*, posture 3 of diagonally sitting), respectively. In contrast, the pixel sizes of pigs are computed as 20,903, 21,770, and 17,695, respectively. Therefore, the size of each pig should be computed in pixel-level accuracy and not in BB-level accuracy. Furthermore, *CB* and *MB* are calculated for each BB detected by TinyYOLO. For a low *CB*, the pigs may not be detected correctly because of the possibility of touching pigs; however, for a high *CB*, the pigs can be detected accurately. In other words, we can compute the size of each pig accurately, if *CB* is larger than the threshold value (*i.e.*, T_3). In this study, we use both *CB* and *MB* to detect the separated pigs without the complicated cases. That is, we select the BB with a high *CB*, and then check *MB*. If *MB* is less than the threshold value (*i.e.*, T_4), the detected BB is discarded.

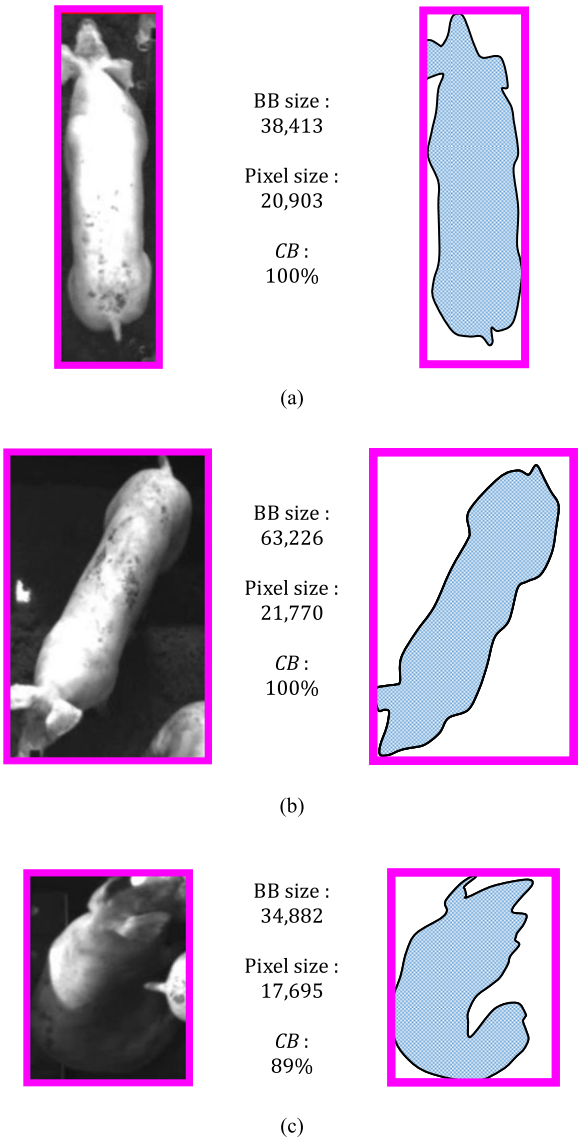


FIGURE 6. Illustration of the difference between the BB and pixel sizes for various postures and directions of the same pig. (a) Posture 1, (b) posture 2, and (c) posture 3.

More specifically, in the post-processing module, we decide whether each RoI can be used to compute the pig size with the help of the constraints on the posture and separability, *i.e.*, *AB, CB,* and *MB*. Note that, *CB* is calculated in the mid-processing module, whereas *AB* and *MB* are calculated in the post-processing module. Figure 7 illustrates how a reliable RoI is decided using *AB*, *CB*, and *MB*. In Figure 7 (a), the RoI corresponding to the current BB is selected as a reliable RoI, because there are sufficient pixel changes within the BB with a high *CB* (*i.e.*, $AB > T_2$ and $CB > T_3$), and the two pigs are separated enough (*i.e.*, $MB > T_4$). In contrast, if any *AB*, *CB*, or *MB* is less than the threshold (*i.e.*, T_2, T_3, and T_4, respectively), the RoI is not selected (*e.g.*, Figure 7 (b) where $MB < T_4$). After selecting the reliable RoI, the size of the pig inside the RoI is computed. Note that, the post-processing module requires a relatively small computational workload. Furthermore, it is executed

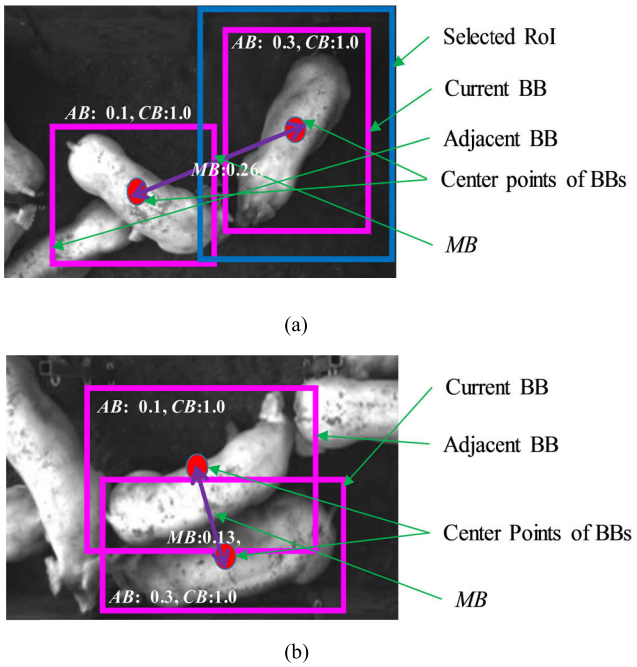


FIGURE 7. Illustration of the reliable RoI decision using *AB*, *CB*, and *MB*. (a) Selected RoI, and (b) discarded RoI.

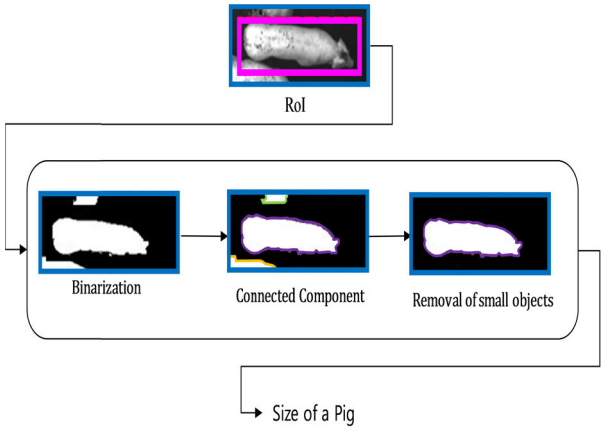


FIGURE 8. Post-processing module for computing the size of each pig.

only when *AF*, *AB*, *CB*, and *MB* are larger than T_1, T_2, T_3, and T_4, respectively. Thus, the post-processing module can be executed on an embedded CPU cost effectively.

C. COMPUTATION OF THE SIZE OF EACH PIG

Finally, the size of each pig inside the reliable RoI is computed. Figure 8 shows the procedure for computing the size of each pig by using image processing techniques such as the binarization and the connected component analysis. In the reliable RoI, the infrared image is transformed into a binary image, each connected component is analyzed to remove the noise (*i.e.*, relative small components), and then, the size of each pig is computed. In general, the connected component analysis requires a relatively large computational workload to segment the foreground pixels. In our method, because the connected component analysis is applied to a small RoI corresponding to the moving pig, it can be executed an embedded CPU.

Algorithm 1 Computing the Size of Each Pig

Input: A video frame from a video stream
Output: Size of each pig in the frame
/* Pre-processing on a CPU */
Step 1: In_frame = Get_frame(Video stream);
GMM_result = GMM(In_frame);
If(AF(GMM_result) > T_1) **GO to Step 2**
ELSE **GO to Step 1**
/* Mid-processing on a GPU */
Step 2: BBs = TinyYOLO(In_frame)}
/* Post-processing on a CPU */
Step 3: FOR(i = 1 to all BBs){
IF(AB(BB_i) > T_2 & CB(BB_i) > T_3 &
MB(BB_i) > T_4){
RoI_i = Clipping(In_frame, BB_i)
$\text{Binary\_img}_\text{i}$ = Binaization(RoI_i)
Seg_img_i = Max_CC($\text{Binary\_img}_\text{i}$)
Pig_size_i = Amount_pixels(Seg _img_i)
RECORD Pig_size_i }}
GO to Step 1

Algorithm 1 summarizes the overall procedure of the image processing and deep learning modules. First, a video frame is obtained from the continuous video stream, and then, GMM is used to check for moving frames in step 1 (*i.e.*, pre-processing module). That is, if *AF* of the GMM result is larger than T_1, only then the mid-processing module is executed. In step 2 (*i.e.*, mid-processing module), individual pigs are detected by TinyYOLO with BB information (*i.e.*, the position coordinates of each BB). In step 3 (*i.e.*, post-processing module), each BB is checked for reliable RoIs. That is, if *AB*, *CB*, and *MB* of each BB are larger than T_2, T_3 and T_4, respectively, only then the size of the pig in the BB (and hence the corresponding RoI) is computed. To compute the size of each pig inside a reliable RoI, the image processing techniques such as binarization (*i.e.*, Otsu) and segmentation (*i.e.*, connected component analysis) are applied. Although Otsu's method is simple, it works well for a pig room captured by a top-view camera (*i.e.*, simple background). Finally, the possible noise is removed, and then the size of the pig in the RoI is computed.

D. PIPELINING CPU AND GPU

By pipelining the CPU and GPU within an embedded device in the continuous video stream, we can overlap the image processing and deep learning modules, and thus, the total execution time can be reduced on the embedded device. The overall procedure for the undergrown pig detection consists of the pre-processing, mid-processing, and post-processing modules. However, there is a data dependency between the

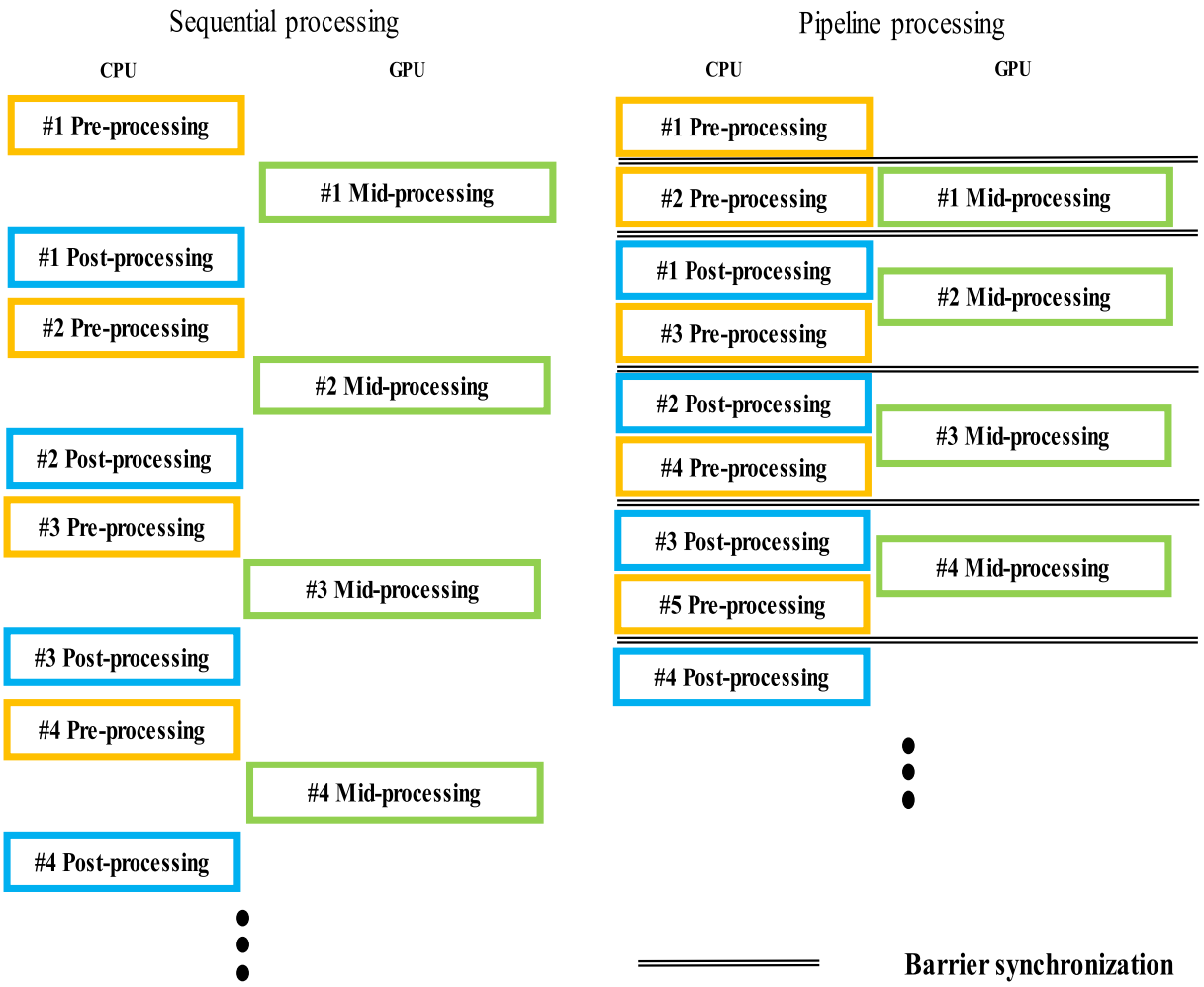


FIGURE 9. Comparison between sequential processing and pipeline processing.

modules for processing a video frame. Therefore, in this study, the modules are executed in a pipelined fashion by overlapping the processing of the previous, current, and next frames under the data dependency.

Figure 9 shows a comparison between the sequential and pipeline processing. Note that, in Figure 9, we illustrate a case where all video frames require all the three modules, for the purpose of simplicity. If the conditions of *AF*, *AB*, *CB*, and *MB* are not satisfied, then the corresponding mid- and/or post-processing modules can be skipped. First, the pre-processing module of #1 frame is executed. After the 1st barrier synchronization, the pre-processing module of #2 frame and the mid-processing module of #1 frame are executed concurrently. Note that, after the 2nd barrier synchronization, (i+1)-th pre-processing, i-th mid-processing, and (i−1)-th post-processing modules are executed concurrently. In order words, the pre- and post-processing modules (executed on the CPU) are overlapped with the mid-processing module (on the GPU) in the pipeline. As shown in Figure 9, the proposed method can reduce the idle time of the sequential processing through the pipelined execution of the continuous video frames across the CPU and GPU.

IV. EXPERIMENTAL RESULTS

A. EXPERIMENTAL ENVIRONMENT AND DATABASES

The experiments for detecting undergrown pigs were conducted on the following embedded device NVIDIA TX-2 (NVIDIA, Santa Clara, CA, USA) [28] under the environment of Dual-core Denver 2 64-bit CPU and quad-core ARM A57 complex, NVIDIA Pascal architecture with 256 NVIDIA CUDA cores, 8 GB 128-bit LPDDR4, and Ubuntu 16.04.2 LTS (Canonical Ltd, London, UK).

We conducted the experiment in two pig rooms of 3.2 m tall and 2.0 m wide × 4.9 m long (denoted as pig room 1, see Figure 10 (a)), and 3.2 m tall and 2.0 m wide × 2.0 m long (denoted as pig room 2, see Figure 10 (b)), respectively, at Chungbuk National University. In pig room 1, there were nine pigs (86.6–98.4 kg) that were 148 days old (born from a sow). As shown in Figure 10 (a), the nine pigs had similar sizes. In pig room 2, there were five pigs (18.7–33.8 kg) that were 62 days old (born from another sow). Note that, in pig room 2, the upper and lower parts of the room were separated by a fence. Because a water pipe was located over the room, a pig captured by the top-view camera could be divided into two body parts, as shown in Figure 10 (b).

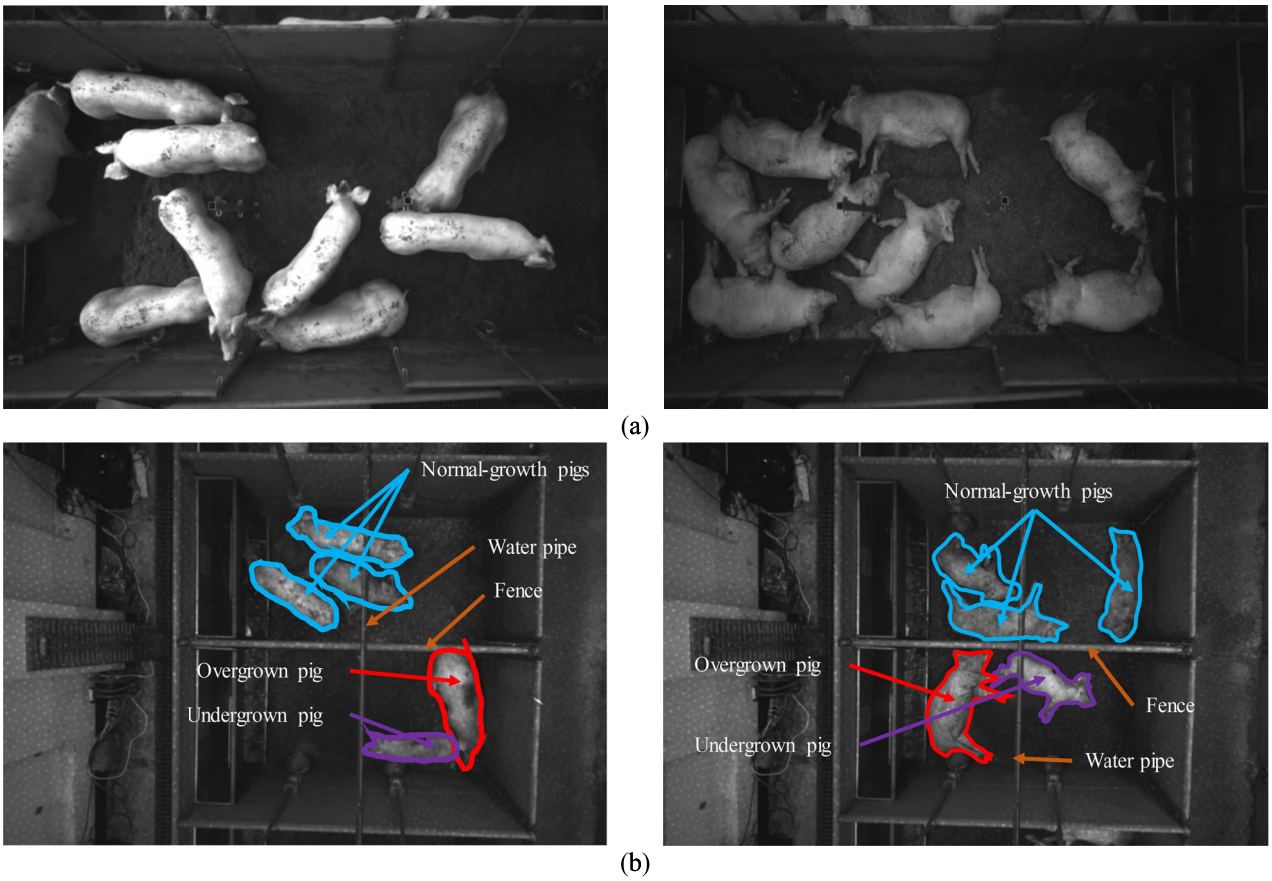


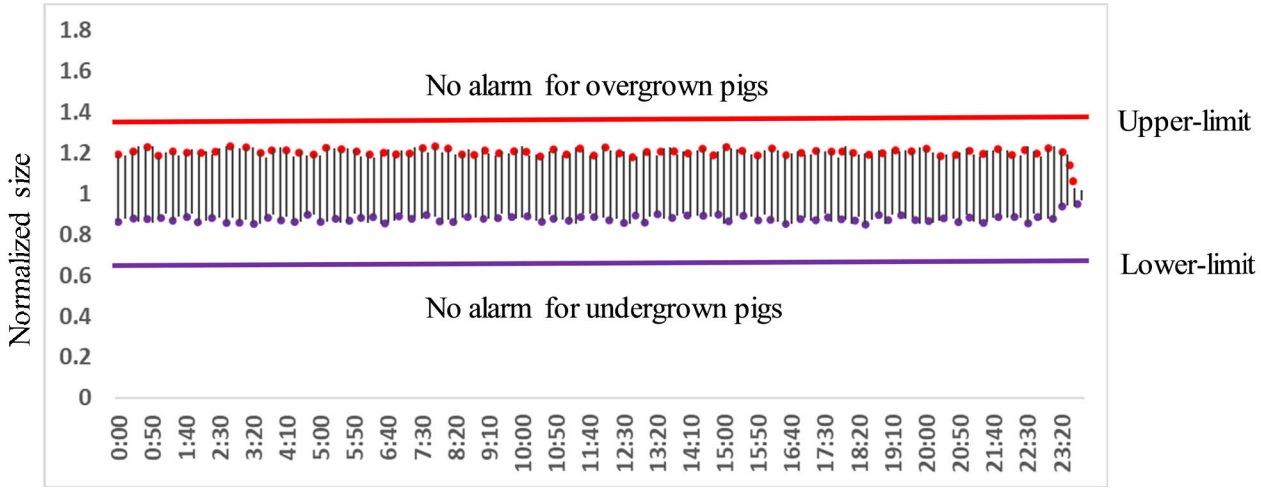
FIGURE 10. The two pig rooms that were monitored. (a) Nine normal-growth pigs (86.6–98.4 kg) in pig room 1. (b) Three normal-growth pigs (24.2–27.8 kg), one overgrown pig (33.8 kg), and one undergrown pig (18.7 kg) in pig room 2.

In addition, the top three pigs (in the upper room) had similar sizes, but the bottom two pigs (in the lower room) were relatively larger and smaller than the top three pigs. Because the five pigs in pig room 2 were born from the same sow (and hence the same ages), we can consider the top three pigs (24.2–27.8 kg), larger pig (33.8 kg), and smaller pig (18.7 kg) as normal-growth pigs, an overgrown pig, and an undergrown pig, respectively. Note that, to avoid the complicated cases (shown on the right side of Figure 10), we considered only the acceptable conditions of posture (*i.e.*, walking pings) and separability, and then computed the relative size of pigs.

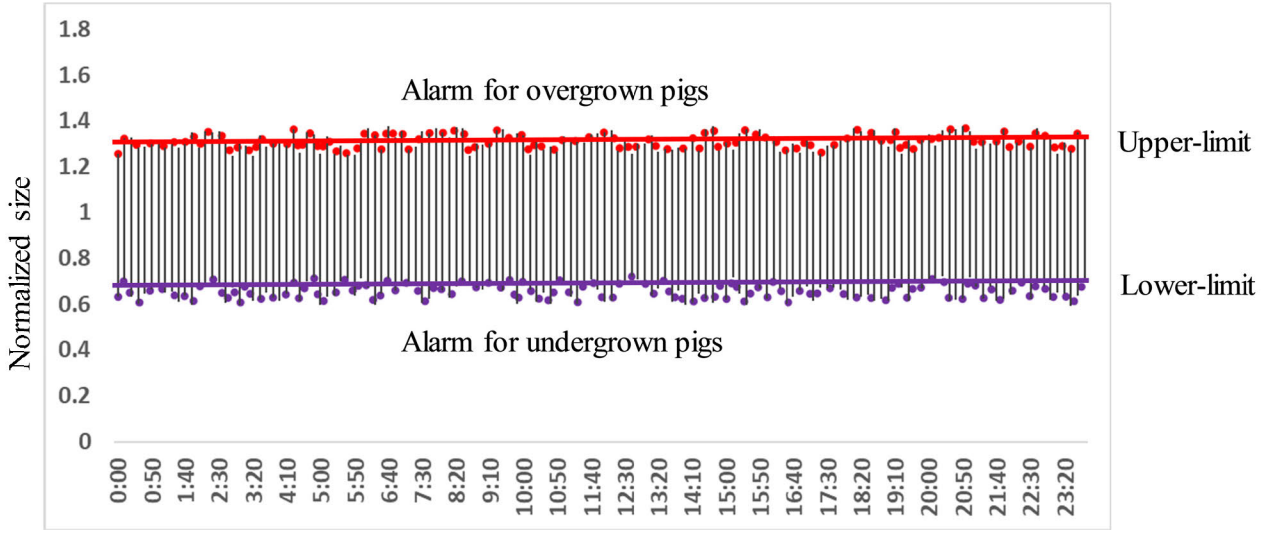
Furthermore, we installed two Intel RealSense cameras (D435 model, Intel, Santa Clara, CA, USA) [35] on the ceiling to obtain the images. We acquired infrared images through the camera, and each image had a resolution of 1280 × 720, 30 frames per second (fps). To analyze the monitoring results for 24 h, ground truth (GT) data was constructed for each 1 min at 10 min intervals in the pig rooms 1 and 2, respectively. In other words, 4,320 frame data for each 1 min (30 fps) were obtained for pig rooms 1 and 2, respectively. Note that, the GT data was manually computed only for moving pigs. Finally, we acquired 3,320 training images and then trained TinyYOLO (with learning rate of 0.001, decay of 0.0005, momentum of 0.5, leaky ReLU as the activation function, default anchor parameter, and the iterations of 20,000). Then, we obtained 1,000 test images and conducted the test with pre-processing, TinyYOLO processing, and post-processing modules. TinyYOLO and image processing steps we used were of TinyYOLO version 3 [36] and OpenCV 3.4 [37], respectively. In addition, we set each threshold (*i.e.*, T_1, T_2, T_3, and T_4) as 0.1, 0.3, 0.9, and 0.2, respectively. Further, we uploaded the source code and demo video to URL (https://github.com/hanseahn/real-time-pig-size-computation-using-pipeline-processing-with-image-and-deep-learning-techniques) for other researches.

B. ANALYSIS OF UNDER-GROWTH MONITORING

Figure 11 shows the detection results of pig's relative size for 24 h with the GT data. As shown in Figure 10, the nine pigs in pig room 1 had similar sizes, while the five pigs in pig room 2 had different sizes. Although all the pigs in each pig room were born from the same sow, the size of each pig might differ depending on several reasons. As explained, in pig room 2, one pig was relatively larger than the other pigs (denoted as an *overgrown pig*) and one pig was relatively smaller than the other pigs (denoted as an *undergrown pig*). We did not attach RFIDs to the pigs, and the pig rooms did not have any facility to control the feeding (*i.e.*, one pig could eat more than



(a)



(b)

FIGURE 11. Detection results with GT data. (a) Detection result of pig room 1. (b) Detection result of pig room 2.

TABLE 2. Solution scenarios for detecting undergrown pigs.

Scenarios	Techniques		Features
S_1 [36]	TinyYOLO version 3	End-to-end deep learning for object detection	BB-level accuracy, Fast
S_2 [39]	Mask R-CNN	End-to-end deep learning for instance segmentation	Pixel-level accuracy, Slow
S_3 (Proposed)	**TinyYOLO + Image processing**	**Combination of deep learning and image processing**	**Pixel-level accuracy, Fast**

the other pigs). It is well known that pigs naturally behave aggressively for social hierarchy status access to resources such as feed [38]. Therefore, the overgrown pig could take over the feed assigned to the undergrown pig. Unfortunately, the undergrown pig died after the video was recorded. This is the reason why we need to detect undergrowth in pigs automatically so that special care can be provided at the right time.

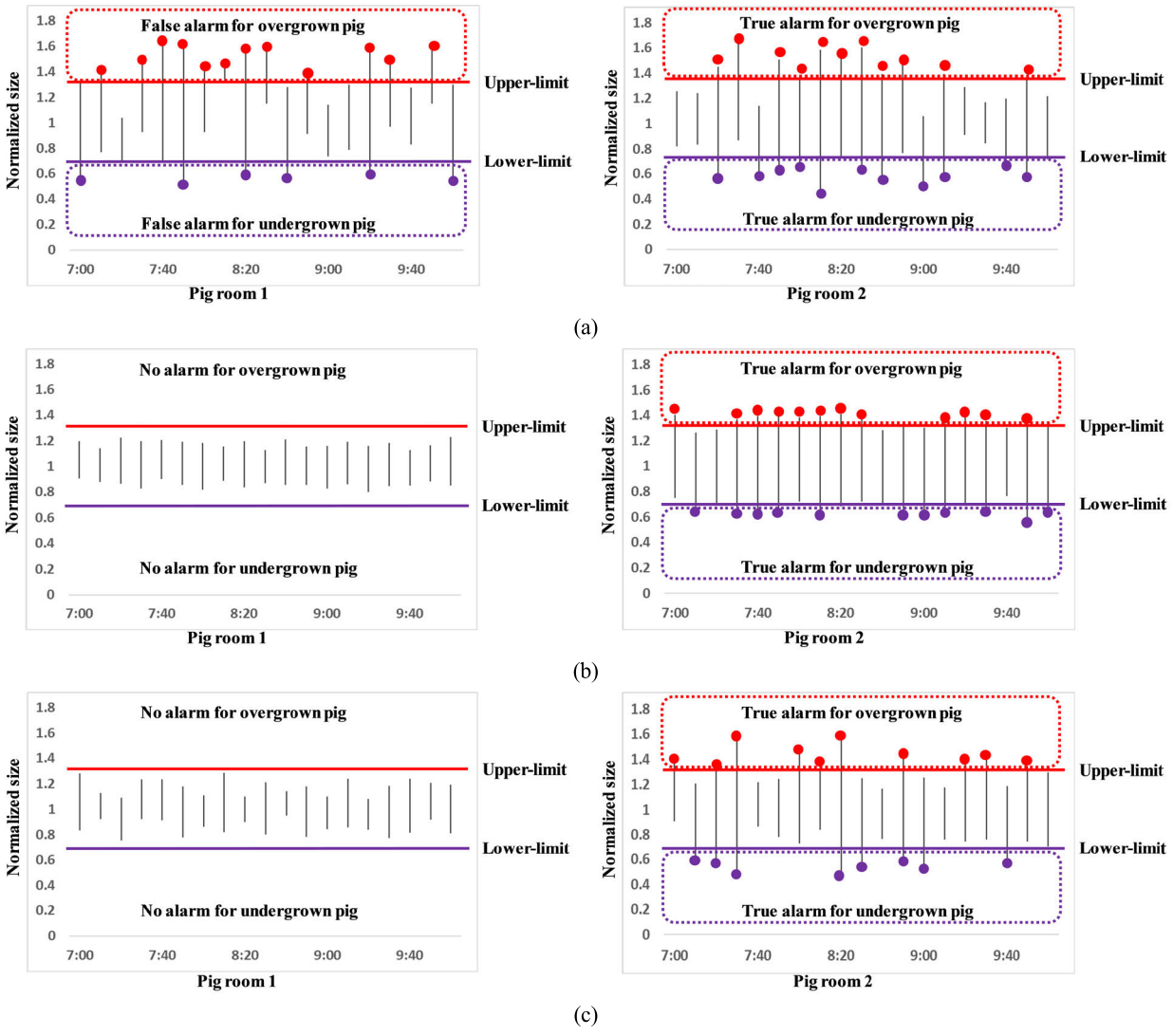


FIGURE 12. Detection results of different solution scenarios. (a) S1: TinyYOLO version 3 [36]. (b) S2: Mask R-CNN [39]. (c) S3: Proposed method.

In Figure 11, the pig's size was normalized with the mean of the pig's size computed for 24 h to represent the relative size of each pig. In pig room 1, all the pig sizes computed from the GT data were in the range between 1.3 (*i.e.*, red line) and 0.7 (*i.e.*, purple line). In other words, in pig room 1, the deviation of pig's size was less than 0.5, because the nine pigs had similar sizes. However, in pig room 2, the overgrown and undergrown pigs were detected over the red line and under the purple line, respectively. In this case, the deviation of pig's size was more than 0.6, because of one overgrown pig and one undergrown pig among the five pigs. Therefore, with the GT data, we can send an alarm to alert the farm worker.

C. ACCURACY ANALYSIS

To evaluate the effectiveness of the proposed method, we considered three solution scenarios for the undergrowth monitoring in group-housed pig rooms. Note that, we used the relative size of each moving pig to detect overgrowth and undergrowth in pigs. In addition, we did not consider the time series analysis techniques (*i.e.*, RNN and LSTM), but used the object detection techniques to detect the individual pigs and compute the relative size of pigs within the video. The first scenario (denoted as S_1) tried to compute the size of each pig by using TinyYOLO only. Because TinyYOLO is a very fast deep learning-based object detector, it can detect each pig on a TX-2. However, because it produces an enclosing BB for each pig, the size of the detected pig may not be accurate. The second scenario (denoted as S_2) tried to compute the size of each pig by using Mask R-CNN [39]. Note that, Mask R-CNN is one of the most widely used instance segmentors based on deep learning techniques. Because Mask R-CNN is a very accurate instance segmentor, it can detect each pig in pixel-level accuracy. However, because it decides each pixel for pig detection, it requires a large computational overhead. Note that, both scenarios S_1 and S_2 rely on an end-to-end deep learning technique only, without any image processing

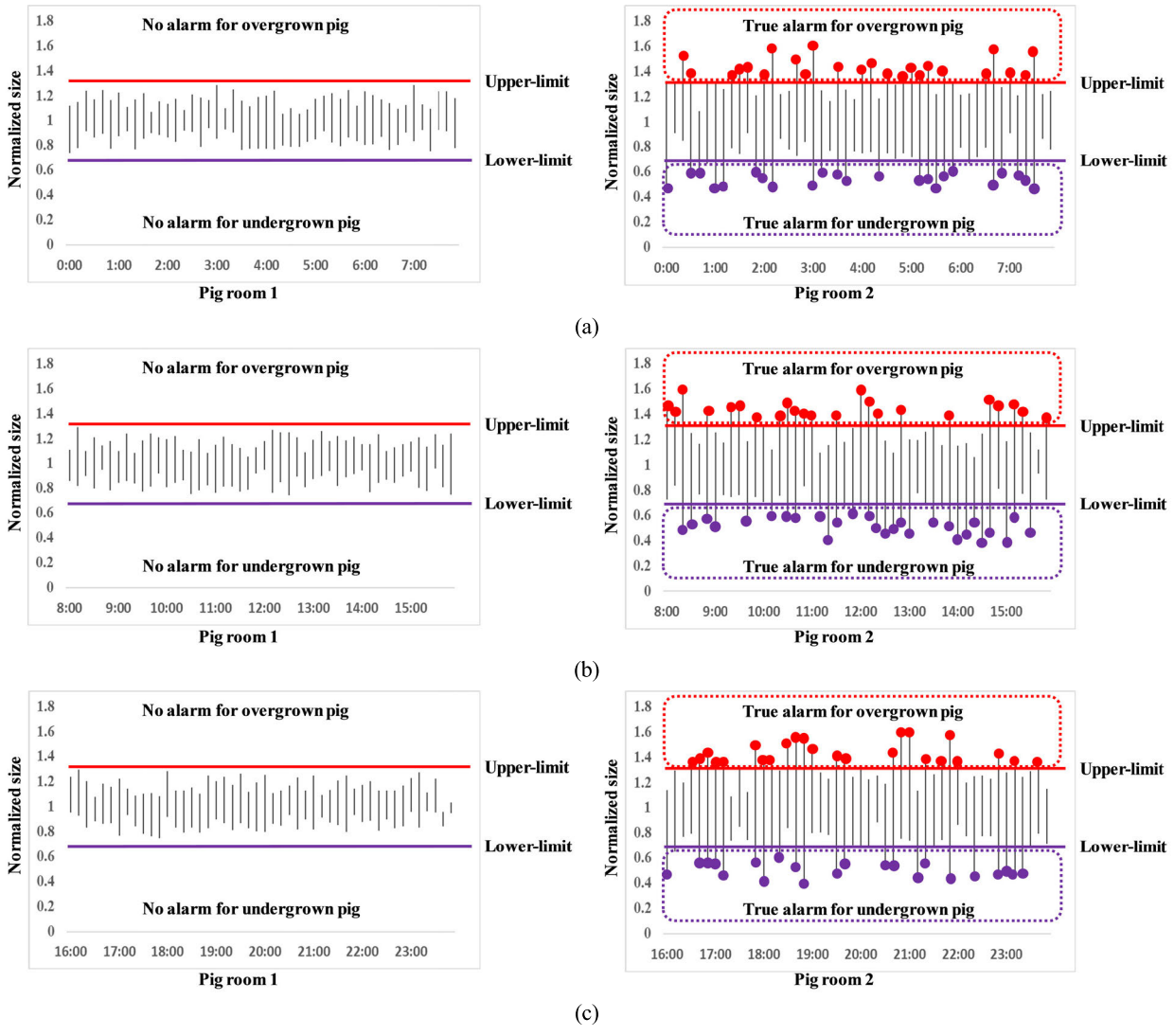


FIGURE 13. Detection results of proposed method (*i.e.*, S_3) for 24 h. (a) 00:00–08:00, (b) 08:00–16:00, and (c) 16:00–00:00.

technique. The third scenario (denoted as S_3) tried to compute the size of each pig by using the proposed method. Because the proposed method combines the deep learning and image processing techniques effectively, it can simultaneously satisfy the accuracy and real-time execution requirements for the undergrown pig monitoring. Note that, for the group-housed pig rooms, developing a ''markerless'' instance segmentor based on image processing techniques only is a difficult task (see Table 1). Therefore, we did not consider the solution scenario that uses image processing techniques only. The features of each solution scenario are summarized in Table 2.

The proposed method computed the sizes of walking and separated pigs and sent an alarm if there were any undergrown or overgrown pigs in a room. Therefore, we verified the proposed method by comparing the alarms generated by the proposed method with those generated by GT. Figure 12 shows the results of monitoring for 3 h (07:00 to 10:00) with three solution scenarios in pig room 1 and pig room 2, respectively. In S_1, the deviation of pig's size was more than 1.0 in both rooms 1 and 2 as shown in Figure 12 (a). Because the size of BB depends on the pig's posture, it was difficult to compute the accurate size of each pig, without any segmentation technique. In the case of S_2, Mask R-CNN provided small deviations similar to GT (see Figure 12 (b)). Thus, we confirmed that Mask R-CNN could compute the accurate size of each pig and detect both overgrown and undergrown pigs. Note that, because Mask R-CNN could not be executed on a TX-2 towing to memory problem, we measured the accuracy of Mask R-CNN on a PC. Finally, in S_3, the deviations of pig's size were 0.6 and 1.1 for the pig rooms 1 and 2, respectively (see Figure 12 (c)). Although S_3 provided a higher deviation (*i.e.*, 0.6) than S_2 (*i.e.*, 0.5), S_3 could detect the overgrown and undergrown pigs automatically.

The detection results of the proposed method for 24 h are shown in Figure 13. Note that, the alarms will be generated

depending on the setting of the upper-limit (*i.e.*, the red line shown in Figure 13) and lower-limit (*i.e.*, the purple line shown in Figure 13). For an upper-limit of 1.3 and a lower-limit of 0.7, S_3 did not generate any false alarms for pig room 1. In addition, for pig room 2, S_3 generated true alarms for the undergrown pig. Of course, S_3 did not generate true alarms for pig room 2 for every time interval. These cases may be considered as the failure cases. When compared with the GT, S_3 cannot compute the size of the undergrown pig perfectly. Further, if the undergrown pig is not in a proper position (*i.e.*, either resting or sleeping) during the time interval, then S_3 will not try to compute its size in that time interval. However, S_3 could generate a sufficient number of true alarms even though it could not compute the size of the undergrown pig perfectly. Therefore, we confirm that the proposed method can detect undergrown pigs automatically without any human inspection or measurement of actual weight by a farm worker.

TABLE 3. Execution time of each module of S_3 scenario measured on a TX-2.

Modules		Average execution time per a frame (ms)
Pre-processing	Get_frame	16.9
	GMM with OpenMP	12.4
Mid-processing	TinyYOLO	24.1
Post-processing	Clipping RoI	2.3
	Computing pig's size	3.4
Total		59.1

D. SPEED ANALYSIS

With a careful combination of image processing and deep learning techniques, we could compute the relative size of each pig on an embedded device TX-2. To analyze the speed performance of the proposed method, the execution time of each module was measured on a TX-2 as shown in Table 3. GMM required the highest execution time in image processing modules, and thus, we parallelized GMM with OpenMP to reduce the execution time by using a multi-core CPU. Note that, the latency (*i.e.*, execution time for a video frame) of the proposed method was 59.1 ms on a TX-2. However, the execution times of the image processing modules could be hidden by pipelining the CPU and GPU steps while processing the continuous video stream.

Finally, Table 4 shows the throughput (*i.e.*, frames per second for a video stream) of the three scenarios on a TX-2. In S_1, the average throughput was 24.3 fps on a TX-2. However,

TABLE 4. Comparison of throughputs measured on a TX-2.

Solution scenarios	Average throughput (fps)
S_1: TinyYOLO version 3 [36]	24.3
S_2: Mask R-CNN [39]	Not executable
S_3: Proposed method	**31.5**

the average throughput of S_3 was 31.5 fps. By pipelining the tasks of a CPU and GPU on a TX-2, we confirmed that the real-time processing was possible. As explained, the average throughput of S_2 could not be measured on a TX-2. To compare the relative speed of the three scenarios, we measured the average throughput of each scenario on a PC (*i.e.*, Intel Core i7-7700K CPU and NVIDIA GeForce GTX 1080 Ti GPU). The average throughputs of the three scenarios were 125.4 fps (S_1), 6.5 fps (S_2), and 161.7 fps (S_3), respectively. The throughput of S_2 was lower than that of S_3 by a factor of 20.1, and thus, S_2 could not be executed on a TX-2 in real-time (even without the memory problem).

As aforementioned, the nine pigs in pig room 1 had similar sizes (*i.e.*, alarm should not be generated), whereas the five pigs in pig room 2 had different sizes (*i.e.*, alarm should be generated). Bounding box-based TinyYOLO could be executed on a TX-2 in real-time (because of the light computational requirement), but the size of each pig detected might not be accurate (*i.e.*, it generated several false alarms for pig room 1, as shown in Figure 12 (a)). In addition, the size of each pig detected by using Mask R-CNN might be accurate (*i.e.*, it did not generate any false alarms for pig room 1 and generated true alarms for pig room 2, as shown in Figure 12 (b)), but it could not be executed on a TX-2 (because of the heavy computational requirement). Thus, it was confirmed that because the proposed method combines the deep learning and image processing techniques effectively, it could simultaneously satisfy the accuracy (*i.e.*, it did not generate any false alarms for pig room 1 and generated true alarms for pig room 2, as shown in Figure 12 (c)) and real-time execution requirements for undergrown pig monitoring.

V. CONCLUSION

Detection of undergrown pigs in group-housed pig rooms is important in pig management because it allows early detection of health and management problems. In this paper, we proposed an automated method for detecting undergrown pigs, which is practically impossible by few farm workers in a large-scale farm. By using the top-view camera installed at the ceiling of a pig room, we computed the size of each moving pig with computer vision techniques. With a careful combination of image processing and deep

learning techniques, we could compute the relative size of each pig on an embedded device TX-2. Furthermore, by pipelining the CPU and GPU steps while processing the continuous video stream, we could hide the additional image processing time for the real-time throughput.

From the experimental results obtained from pig room 1 (consisting of nine pigs of similar sizes), it was observed that the proposed method did not generate a false alarm for undergrown pigs. Although Mask R-CNN did not generate a false alarm owing to the capability of pixel-level accuracy, it could not be executed on a TX-2 because of the memory problem. However, TinyYOLO generated a false alarm owing to the limitation of BB-level accuracy.

From the experimental results obtained from pig room 2 (consisting of three pigs of similar sizes, one pig of larger size, and one pig of smaller size), the proposed method could generate a true alarm for undergrown pigs in real time. Similar to pig room 1, TinyYOLO generated an alarm, but it was meaningless because of the possibility of a false alarm. Mask R-CNN, owing to its pixel-level accuracy could generate a true alarm, but it was meaningless because of the memory problem.

In summary, the recent developments in deep learning techniques can be applied to several computer vision tasks. When we apply such end-to-end deep learning techniques to a complicated task (*i.e.*, instance segmentation) on an embedded device, we need to consider its computational overhead and/or memory constraint. However, by combining the very fast image processing and deep learning techniques carefully, we can perform any complicated task on an embedded device in real-time. That is, IoT-based monitoring techniques for smart pig farms can be applied to intelligent services such as early warning, in addition to simple services such as temperature/humidity control. To the best of our knowledge, this is the first study to detect undergrown pigs in group-housed rooms on a TX-2, without any human intervention or a special facility. However, the proposed method used for detecting the abnormality in growth rate automatically should be considered as an aid (*i.e.*, sending alarms) rather than as a replacement for farm workers.

This proposed method can be extended to other complicated tasks that are related with IoT-based monitoring in a cost-effective way, although we developed a method for detecting undergrown pigs. In addition, we expect that the individual pigs can be managed more effectively if the proposed method is combined with the future work that involves 24-h tracking of each pig (*i.e.*, our final goal for intelligent pig monitoring).

REFERENCES

[1] E. Vranken and D. Berckmans, "Precision livestock farming for pigs," *Animal Frontiers*, vol. 7, no. 1, pp. 32–37, 2017.

[2] H. Kim, S. Jeong, and H. Yoe, "Design and implementation of ICT-based system for information management of livestock farm," *Int. J. Smart Home*, vol. 8, no. 2, pp. 1–6, 2014.

[3] J. Lee, L. Jin, D. Park, Y. Chung, and H.-H. Chang, "Acoustic features for pig wasting disease detection," *Int. J. Inf. Process. Manage.*, vol. 6, no. 1, p. 37, 2015.

[4] J. Lee, L. Jin, D. Park, and Y. Chung, "Automatic recognition of aggressive behavior in pigs using a Kinect depth sensor," *Sensors*, vol. 16, no. 5, p. 631, 2016.

[5] M. Ju, Y. Choi, J. Seo, J. Sa, S. Lee, Y. Chung, and D. Park, "A Kinect-based segmentation of touching-pigs for real-time monitoring," *Sensors*, vol. 18, no. 6, p. 1746, 2018.

[6] S. Mohanty and S. Vyas, "Decentralized autonomous organizations = blockchain + AI + IoT," in *How to Compete in the Age of Artificial Intelligence*, 1st ed. New York, NY, USA: Apress, 2018, pp. 189–206.

[7] M. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," 2019, *arXiv:1908.00080*. [Online]. Available: https://arxiv.org/abs/1908.00080

[8] M. Ham, J. J. Moon, G. Lim, W. Song, J. Jung, H. Ahn, S. Woo, Y. Cho, J. Park, S. Oh, H.-S. Kim, "NNStreamer: Stream processing paradigm for neural networks, toward efficient development and execution of on-device AI applications," 2019. *arXiv:1901.04985*. [Online]. Available: https://arxiv.org/abs/1901.04985

[9] N. Mowla, I. Doh, and K. Chae, "On-device AI-based cognitive detection of bio-modality spoofing in medical cyber physical system," *IEEE Access*, vol. 7, pp. 2126–2137, 2018.

[10] N. Brandl and E. Jorgensen, "Determination of live weight of pigs from dimensions measured using image analysis," *Comput. Electron. Agricult.*, vol. 15, no. 1, pp. 57–72, 1996.

[11] *Nedap Livestock Management*. [Online]. Available: http://en.nedap-livestockmanagement.com

[12] C. P. Schofield, "Evaluation of image analysis as a means of estimating the weight of pigs," *J. Agricult. Eng. Res.*, vol. 47, pp. 287–296, Sep./Dec. 1990.

[13] J. Wu, R. Tillett, N. Mcfarlane, X. Ju, J. Siebert, and P. Schofield, "Extracting the three-dimensional shape of live pigs using stereo protogrammetry," *Comput. Electron. Agricult.*, vol. 44, no. 2, pp. 203–222, 2004.

[14] T. M. Banhazi, M. Tscharke, W. M. Ferdous, C. Saunders, and S. Lee, "Improved image analysis based system to reliably predict the live weight of pigs on farm: Preliminary results," *Austral. J. Multi-Disciplinary Eng.*, vol. 8, no. 2, pp. 107–119, 2011.

[15] A. Wongsriworaphon, S. Pathumnakul, and B. Arnonkijpanich, "Image analysis for pig recognition based on size and weight," in *Proc. IEEE Int. Conf. Ind. Eng. Manage.*, Hong Kong, Dec. 2012, pp. 856–860.

[16] J. Kongsro, "Estimation of pig weight using a Microsoft Kinect prototype imaging system," *Comput. Electron. Agricult.*, vol. 109, pp. 32–35, Nov. 2014.

[17] A. Wongsriworaphon, B. Arnonkijpanich, and S. Pathumnakul, "An approach based on digital image analysis to estimate the live weights of pigs in farm environments," *Comput. Electron. Agricult.*, vol. 115, pp. 26–33, Jul. 2015.

[18] C. Shi, G. Teng, and Z. Li, "An approach of pig weight estimation using binocular stereo system based on LabVIEW," *Comput. Electron. Agricult.*, vol. 129, pp. 37–43, Nov. 2016.

[19] I. Condotta, T. Brown-Brandl, K. Silva-Miranda, and J. Stinn, "Evaluation of a depth sensor for mass estimation of growing and finishing pigs," *Biosys. Eng.*, vol. 173, pp. 11–18, Sep. 2018.

[20] A. Pezzuolo, M. Guarino, L. Sartori, L. A. González, and F. Marinello, "On-barn pig weight estimation based on body measurements by a Kinect v1 depth camera," *Comput. Electron. Agricult.*, vol. 148, pp. 29–36, May 2018.

[21] A. Pezzuolo, V. Milani, D. Zhu, H. Guo, S. Guercini, and F. Marinello, "On-barn pig weight estimation based on body measurements by structure-from-motion (SfM)," *Sensors*, vol. 18, no. 11, p. 3603, 2018.

[22] K. Jun, S. J. Kim, and H. W. Ji, "Estimating pig weights from images without constraint on posture and illumination," *Comput. Electron. Agricult.*, vol. 153, pp. 169–176, Oct. 2018.

[23] K. Wang, D. Zhu, H. Guo, Q. Ma, Q. Su, and Y. Su, "Automated calculation of heart girth measurement in pigs using body surface point clouds," *Comput. Electron. Agricult.*, vol. 156, pp. 565–573, Jan. 2019.

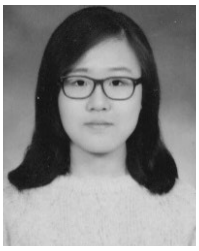
[24] M. A. Kashiha, C. Bahr, S. Ott, C. P. H. Moons, T. A. Niewold, F. O. Ödberg, and D. Berckmans, "Automatic weight estimation of individual pigs using image analysis," *Comput. Electron. Agricult.*, vol. 107, pp. 38–44, Sep. 2014.

[25] M. Kashiha, C. Bahr, S. Ott, C. P. H. Moons, T. A. Niewold, F. O. Ödberg, and D. Berckmans, "Weight estimation of pigs using top-view image processing," in *Proc. Int. Conf. Image Anal. Recognit.*, Algrave, Portugal, 2014, pp. 496–503.

[26] D. Jensen, K. Dominiak, and L. Pedersen, "Automatic estimation of slaughter pig live weight using convolutional neural networks," in *Proc. 2nd Int. Conf. Agro BigData Decis. Support Syst. Agricult.*, At Lleida, Spain, 2018, pp. 1–4.
[27] M. Lu, T. Norton, A. Youssef, N. Radojkovic, A. P. Fernández, and D. Berckmans, "Extracting body surface dimensions from top-view images of pigs," *Int. J. Agricult. Biol. Eng.*, vol. 11, no. 5, pp. 182–191, 2018.
[28] NVIDIA. Accessed: Apr. 30, 2019. *NVIDIA Jetson TX2*. [Online]. Available: http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html
[29] S. Lee, H. Kim, J. Sa, B. Park, and Y. Chung, "Real-time processing for intelligent-surveillance applications," *IEICE Electron. Exp.*, vol. 14, no. 8, 2017, Art. no. 20170227.
[30] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 7236–7271.
[31] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.
[32] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.
[33] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1440–1448.
[34] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[35] Intel. *Intel RealSense D435*. Accessed: Feb. 28, 2018. [Online]. Available: https://click.intel.com/intelr-realsensetm-depth-camera-d435.html
[36] AlexeyAB. *Darknet*. Accessed: Apr. 30, 2019. [Online]. Available: https://github.com/AlexeyAB
[37] OpenCV. *Open Source Computer Vision*. Accessed: Apr. 30, 2019. [Online]. Available: http://opencv.org
[38] R. D'Eath and S. Turner, "The natural behaviour of the pig," in *The Welfare Pigs*. Dordrecht, The Netherlands: Springer, 2009, pp. 13–45.
[39] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2961–2969.

SUNGJU LEE received the B.S., M.S., and Ph.D. degrees in computer science from Korea University, South Korea, in 2006, 2008, and 2012, respectively. He joined Korea University—Sejong, in 2015, where he is currently an Assistant Professor with the Department of Computer Convergence Software. His research interests include video surveillance, object detection, deep network optimization, and IT convergence.

HANSE AHN received the B.S. degree in computer science from Korea University—Sejong, South Korea, in 2019, where he is currently pursuing the M.S. degree. His current research interests include video surveillance, object detection, and deep network optimization.



JIHYUN SEO received the B.S. degree in computer science from Korea University—Sejong, South Korea, in 2019, where she is currently pursuing the M.S. degree. Her current research interests include video surveillance, object detection, and deep network optimization.

YONGWHA CHUNG received the B.S. and M.S. degrees in electronic engineering from Hanyang University, South Korea, in 1984, and the Ph.D. degree in computer science from the University of Southern California, USA, in 1997. He was a Team Leader with ETRI, from 1986 to 2003. He joined Korea University—Sejong, in 2003, where he is currently a Professor with the Department of Computer Convergence Software. His research interests include parallel processing, image processing, and IT convergence.

DAIHEE PARK received the B.S. degree in mathematics from Korea University, South Korea, in 1982, and the Ph.D. degree in computer science from Florida State University, USA, in 1992. He joined Korea University—Sejong, in 1993, where he is currently a Professor with the Department of Computer Convergence Software. His research interests include big data, data mining, deep learning, and IT convergence.

SUNGBUM PAN received the B.S., M.S., and Ph.D. degrees in electronics engineering from Sogang University, South Korea, in 1991, 1995, and 1999, respectively. He was a Team Leader with the Biometric Technology Research Team, ETRI, from 1999 to 2005. He is currently a Professor with Chosun University. His current research interests include biometrics, security, and VLSI architectures for real-time image processing.