Heterogeneous Computing for a Real-Time Pig Monitoring System
Younchang Choi*, Jinseong Kim, Jaehak Kim, Yeonwoo Chung, Yongwha Chung, Daihee Park, and Hakjae Kim
aDept. of Computer and Information Science, Korea Univ., 2511 Sejong Ave., Sejong-si, Republic of Korea 30019;
bClassAct, 130 Digital Ave., Geumcheon-gu, Seoul, Republic of Korea 08589

ABSTRACT
Video sensor data has been widely used in automatic surveillance applications. In this study, we present a method that automatically detects pigs in a pig room by using depth information obtained from a Kinect sensor. For a real-time implementation, we propose a means of reducing the execution time by applying parallel processing techniques. In general, most parallel processing techniques have been used to parallelize a specific task. In this study, we consider parallelization of an entire system that consists of several tasks. By applying a scheduling strategy to identify a computing device for each task and implementing it with OpenCL, we can reduce the total execution time efficiently. Experimental results reveal that the proposed method can automatically detect pigs using a CPU-GPU hybrid system in real time, regardless of the relative performance between the CPU and GPU.

Keywords: Agriculture IT, Computer Vision, Parallel Processing

1. INTRODUCTION

Several studies have been recently conducted that use surveillance techniques to manage pigs automatically in what are known as “smart pig farms” [1]. A typical Korean pig farm consists of 20 to 30 pigs that are grouped together and managed in a pig room by farm administrators. Several sensors such as gyro and radio frequency identification (RFID) tags are used to automate the management of the farm. However, these approaches increase costs and require additional manual labor for activities. Studies that analyze video sensor data for pig management have been conducted [2-5] in order to assess and prevent these problems.

In this study, we focus on automatic temperature control by employing a video sensor. In particular, caring for (i.e., assessing and controlling the thermal comfort of) weaning pigs (21–28 days old) is the most crucial aspect of pig management because of their weak immunity. This focus on pig comfort in pig housing is typically based on predetermined ambient temperature levels. However, this traditional approach cannot meet a pig’s true thermal needs because it does not consider other factors such as draft, humidity, radiation, floor type and condition. To solve this problem, computer vision-based solutions [6,7] have been proposed to assess pig thermal comfort. These solutions employ image analysis to assess the resting behavior of group-housed pigs. However, these solutions use gray-level image information and thus cannot work in low- to no-light conditions such as at night.

Recently, inexpensive depth sensors such as Microsoft Kinect have been released. In this study, we use this depth sensor to solve the assessment problem of pig thermal comfort at night. We first apply some noise reduction techniques because Kinect produces considerable noise. We then perform background subtraction and binarization to detect pigs in a pig room. For a real-time implementation, we also propose a means of reducing execution time by applying parallel processing techniques. In general, most parallel processing techniques are used to parallelize a specific task. In this study, we consider parallelization of an entire vision system that consists of several tasks. By measuring the speed of each task in a computing device (i.e., CPU or GPU), we can determine an appropriate computing device for each task and implement it using OpenCL. We can then reduce the total execution time efficiently. To the best of our knowledge, this is the first study that automatically detects pigs at night on a CPU-GPU hybrid computing system in real time.

*ycc4477@korea.ac.kr; phone +82 044-860-1343;
2. PROPOSED APPROACH

2.1 Thermal Comfort Assessment System

Ideally, foreground (i.e., pig) detection should be very simple using a depth sensor because depth data are unaffected by illumination changes and color. Practically, however, many problems remain in differentiating between background and foreground. For example, in a pig room we monitored, the floor is a plastic slat with holes for excreta treatment. With a time-of-flight (ToF) based Kinect Version 2 sensor, this floor structure produces many holes (i.e., noise). Furthermore, the Kinect sensor has maximum distances (i.e., 4.5 m) and fields-of-view (i.e., 70.6° horizontally and 60° vertically) at which it can detect depth. Therefore, the depth value returned from a wall may have unreliable values. To address these noise and unreliable values, all depth values below a certain threshold are discarded and both spatial and temporal interpolations are applied by reducing the resolution size and frame rate. Following spatiotemporal interpolation, the pixel values of the background subtraction that are greater than a threshold are regarded as foreground. Morphological operators are then applied to smooth the detected foreground. Finally, each area of the connected component labeling that is greater than the area of a single pig (i.e., area of touching pigs) is added to assess pig thermal comfort automatically.

2.2 Parallel Scenarios in a CPU-GPU Hybrid Computing System

![Diagram of pig detection on a CPU-GPU hybrid computing system.](image)

Figure 1. Implementation of pig detection on a CPU-GPU hybrid computing system.

In our OpenCL-based system, each task can be assigned to a device (deviceCPU or deviceGPU). However, as shown in Fig. 1, we must consider both the communication and computation times in order to determine the shortest path from the host (start) node to the host (end) node. By measuring the communication and computation times of each task on each platform, we can determine task distribution, which then yields the shortest execution time of the thermal comfort assessment system regardless of the relative performance between CPU and GPU in a given platform.

In particular, we assume that the communication between deviceCPU and deviceGPU goes through the host, as transferring data between the devices directly is impossible. Our system also differentiates the communication overhead from the host to the device (“host-to-device + device-to-host”) and from the device to the host (“0”), allowing us to determine easily the appropriate device for each task. In Fig. 1, “commCPU” means “host-to-deviceCPU + deviceCPU-to-host” and “commGPU” means “host-to-deviceGPU + deviceGPU-to-host.” In addition, the greedy rule that determines the task distribution for each platform is “large-task with large-parallelism first”. “Large-task” means long
sequential time, whereas “large-parallelism” means large speedup (including the communication overhead in computing the parallel time). Thus, by multiplying the sequential time ($T_{seq}$) and speedup ($T_{seq}/T_{parallel}$) for each task and interpreting them as our scheduling metric (denoted as “Efficiency”), we can determine the right device for each task. The details of the greedy algorithm are shown in Fig. 2.

\[
Efficiency[i, \text{worker}] = \frac{\text{Comp}[i, \text{host}]}{\text{Comp}[i, \text{worker}] + \text{Comm}[i, \text{worker}]}
\]

3. EXPERIMENTAL RESULTS

The camera was located 4 m above the floor to monitor a pig room measuring 4 m × 3 m, and 13 weaning pigs were present in the room. In our experiments, we set the resolution size to 256 × 212 pixels and the frame rate to 10 frames per second (fps) to handle the noise and unreliable values. From the images of size 256 × 212, we masked out some regions where pigs could not be detected. A background image was computed as a pixel-by-pixel average of a 10-min video sequence without pigs.

Figure 3. Results of the thermal comfort assessment system.
Fig. 3(a) shows an example of an input image (i.e., Kinect depth image), and Fig. 3(b) shows an output image of the thermal comfort assessment system. First, we used spatiotemporal interpolation to reduce noise and unreliable values. We then extracted the foreground using background subtraction. The remaining noise could be eliminated by using morphological operators, and the area of touching pigs was computed in order to assess thermal comfort automatically.

To evaluate the proposed method, we conducted our experiment using two platforms. The first platform (denoted as “Platform 1”) consisted of a 3.50-GHz Intel® Core™ i5-4690 CPU with four cores, a GeForce GTX 760, and 8 GB of RAM. The second platform (denoted as “Platform 2”) consisted of a 2.67-GHz Intel® Core™ i5 CPU 750 with four cores, a GeForce GTS 250, and 8 GB of RAM. We considered Platforms 1 and 2 as higher-performance GPU and higher-performance CPU platforms, respectively.

By measuring the communication and computation times of each task on each platform, we could derive the optimal solution, regardless of the relative performance between CPU and GPU of a given platform. As shown in Fig. 4, the speedup of the proposed method was better than that of the GPU-only method (i.e. all tasks were executed on deviceGPU), regardless of the platform on which it was used. Note that Task 4 (connected component analysis) was too slow on deviceGPU, although we implemented the method for reducing the execution time of merging intermediate values [10]. Because the number of merge steps on a manycore GPU was much greater than that on a multicore CPU, the computational characteristics of Task 4 did not match well with deviceGPU. Thus, the proposed method can provide better performance than the GPU-only method (i.e. all tasks including Task 4 were executed on deviceGPU).

![Figure 4. Performance comparison of each platform.](image)

4. CONCLUSIONS

By using the depth information obtained from a Kinect sensor, we proposed a method to detect pigs at night automatically on a CPU-GPU hybrid computing system. We reduced the noise in the depth information obtained from a Kinect and then detected pigs using background subtraction. To satisfy the real-time requirement, we parallelized the entire monitoring system by identifying a computing device for each task and implementing it using OpenCL. Experimental results confirmed that the proposed method can provide better performance than the GPU-only method (i.e. all tasks were executed on deviceGPU) by a factor of 3 (on Platform 1) and 10 (on Platform 2), respectively. As a future work, using both data and task parallelism and/or using CPU and GPU simultaneously to further reduce the total execution time.
ACKNOWLEDGMENTS.

This research was supported by Leading Human Resource Training Program of Regional Neo Industry through NRF funded by MSIP (2016H1D5A1910730).

REFERENCES