Pipelined Execution of a Pig Monitoring Application on a Heterogeneous Computing Environment

Y. Chung[1], **Y. Choi**[2], **J. Kim**[2], **J. Kim**[2], **Y. Chung**[2], **D. Park**[2], **and H. Kim**[3]
[1]Department of Applied Statistics, Korea University, Sejong City, Republic of Korea
[2]Department of Computer Convergence Software, Korea University, Sejong City, Republic of Korea
[3]Class Act, Seoul City, Republic of Korea

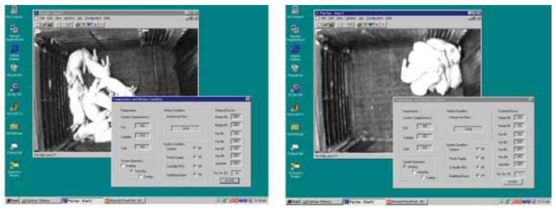
Abstract - *A video sensor data has been widely used for automatic surveillance applications. In this study, we present a method for detecting pigs in a pig room automatically by using the depth information obtained from a Kinect sensor. Especially for a real-time implementation, we propose a way to reduce the execution time by applying parallel processing techniques. In general, most parallel processing techniques have been applied in parallelizing a specific task. In this study, we consider parallelization of the entire system consisting of several tasks. By applying a pipeline scheduling strategy in determining a computing device for each task in order to fully utilize the computational resources of multicore CPU and manycore GPU and implementing it with OpenCL, we can reduce the total execution time efficiently. Based on experimental results, the proposed method can automatically detect pigs using a CPU-GPU heterogeneous computing platform in real time, regardless of the relative performance between CPU and GPU.*

Keywords: Agriculture IT, Computer Vision, Parallel Processing

1 Introduction

Several studies have been recently conducted that use surveillance techniques to automatically manage pigs in what are known as a "smart pig farm" [1]. A typical Korean pig farm consists of 20 to 30 pigs that are grouped together and managed in a pig room overseen by a few farm administrators. Several sensors such as gyro and RFID tags are used to automate the management of the pig farm. However, these approaches increase costs and require additional manual labor for activities. To avoid this, studies that analyze a video sensor data for the pig management have been reported [2-5].

In this study, we focus on the automatic temperature control by using a video sensor. Especially, caring (i.e., assessment and control of thermal comfort) weaning pigs (21 or 28 days old) is the most important issue in the pig management because of their weak immunity. Assessment and control of the environment and thus pig comfort in pig housing is typically based on predetermined ambient temperature levels. However, this traditional approach cannot meet the pig's true thermal needs because it does not integrate the effects of other factors such as drafts, humidity, radiation, floor type/condition. To solve this problem, computer vision-based solutions [6,7] have been proposed in order to assess the thermal comfort with image analysis of resting behavior of group-housed pigs, as shown in Fig. 1. However, these solutions use gray-level image information and thus cannot work in a light-less environment such as night.



(a) Pigs at rest – comfortable (b) Pigs at rest - cold

Fig. 1: Automatic assessment of thermal comfort by using computer vision techniques [7].

Recently, cheap depth sensors such as Microsoft Kinect have been released, and we use this depth sensor to solve the assessment problem of thermal comfort at night. We first apply some noise reduction techniques because Kinect has many noises. Then, we perform background subtraction and binarization to detect pigs in a pig room. For a real-time implementation, we also propose a way to reduce the execution time by applying parallel processing techniques. In general, most parallel processing techniques have been applied in parallelizing a specific task. In this study, we consider OpenCL-based parallelization of the entire vision system consisting of several tasks. By measuring the speed of each vision task in a computing device (i.e., CPU or GPU) and balancing the execution times of CPU and GPU, we can determine a computing device for each task and partition the entire tasks into CPU and GPU. To the best of our knowledge, this is the first pipelined execution that automatically detects pigs at night on a CPU-GPU heterogeneous computing platform in real time.

2 Background

OpenCL [8,9] is an open standard aimed at providing a programming environment suitable to access heterogeneous architectures. In particular, OpenCL (shown in Fig. 2) allows to execute computational workloads on various multicore processors. Considering the increasing availability of such types of processors, OpenCL is playing a crucial role in enabling portable applications to access a wide range of computational resources. To achieve this aim, various levels of abstraction have been introduced in the OpenCL model.

- *Platform* performs an abstraction of the number and type of computing devices in a hardware platform. At this level are made available to developers the routines to query and to manage the computing devices, to create the contexts and work queues for submission of sets of instructions called kernels.
- *Execution* is based on the concept of kernel which is a collection of instructions executed on the computing device, multicore CPU or GPU, called OpenCL device. An OpenCL application can be divided in two programs: host and kernel. The host program is executed on CPU. It defines the context for the kernels and manages their execution. Especially, when a kernel is submitted for execution by the host, an index space is defined. An instance of the kernel executes for each point in this index space. This kernel instance is called a work-item and is identified by its point in the index space, which provides a global ID for the work-item. Each work-item executes the same code on distinguished data. That is, work-items are organized into work-groups providing a more coarse-grained decomposition of the index space.
- *Language* describes the syntax and programming interface for writing kernels (set of instructions that execute on a computing device such as multicore CPUs or GPUs)

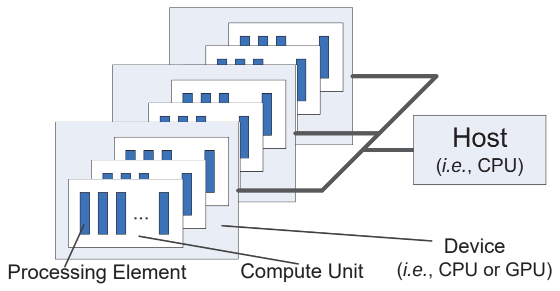


Fig. 2: Platform model of OpenCL

3 Proposed approach

3.1 Thermal Comfort Assessment System

Ideally, foreground (i.e., pig) detection is very simple with a depth sensor because depth data is unaffected by illumination changes and color. Practically, however, there are many problems in differentiating between background and foreground. In a pig room we monitored, for example, the floor is plastic slat with holes for excreta treatment. With a Time-of-Flight (ToF) based Kinect version 2 sensor, this structure of the floor causes many holes (i.e., noise). Furthermore, the Kinect sensor has maximum distances (i.e., 4.5m) and field-of-views (i.e., 70.6o horizontally and 60o vertically) at which it can detect depth. Therefore, the depth value returned from wall may have unreliable values. In order to handle these noise and unreliable values, all depth values below a certain threshold value are discarded and both spatial and temporal interpolations are applied by reducing the resolution size and the frame rate. After the spatiotemporal interpolation, the pixel values of the background subtraction which are larger than a threshold are regarded as foreground. Then, morphological operators are applied to smooth the foreground detected. Finally, each area of the connected component labeling which is larger than the area of one pig (i.e., area of touching-pigs) is added for assessing the thermal comfort automatically.

3.2 Pipelined Execution on a CPU-GPU Heterogeneous Computing Platform

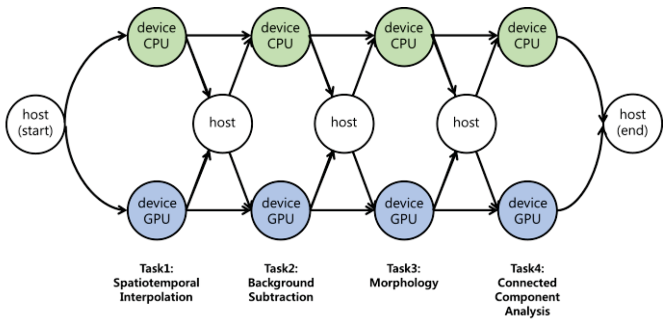

host (start)
device CPU
device GPU
host
host (end)
Task1: Spatiotemporal Interpolation
Task2: Background Subtraction
Task3: Morphology
Task4: Connected Component Analysis

Fig. 3: Possible implementation scenarios in a CPU-GPU heterogeneous computing platform..

In our OpenCL-based system, each task can be assigned to a device (deviceCPU or deviceGPU). As shown in Fig. 3, however, we should consider the communication time as well as the computation time, in order to determine the shortest path from the host (start) node to the host (end) node. Especially, by partitioning the entire tasks into two parts such that the execution time of deviceCPU is balanced with that of deviceGPU, we can maximally utilize the computational resources of multicore CPU and manycore GPU in the thermal comfort assessment system, regardless of the relative performance between CPU and GPU of a given platform.

Especially, it is assumed that the communication between deviceCPU and deviceGPU goes through the host, as it is impossible to transfer data between the devices directly. In addition, the greedy rule that determines the task distribution for each platform is “minimizing the total execution time such that the execution times of CPU and GPU are balanced”. The details of the pipeline scheduling algorithm are shown in Fig. 4.

```
Input   deviceCPU_time[n tasks], deviceGPU_time[n tasks]
Output  Boundary_index, Task_seq(0 = CPU first, 1= GPU first)

Algorithm
Step 1: Initialize Boundary_index, sum1, sum2, Task_seq, total_time
        Boundary_index = 0;
        Define sum of deviceGPU_time[n tasks] as sum1
                and deviceCPU_time[n tasks] as sum2
        if (sum1 > sum2) { Task_seq = 1; total_time = sum2; }
        else { Task_seq = 0; total_time = sum1; }
Step 2: Determine the Boundary_index by increasing the index of tasks
        sum3 = sum4 = 0;
        for i = 1 to n {
            sum1 = sum1 - deviceGPU_time[i-1];
            sum2 = sum2 - deviceCPU_time[i-1];
            sum3 = sum3 + deviceCPU_time[i-1];
            sum4 = sum4 + deviceGPU_time[i-1];
            tmp_time = Min[Max(sum1, sum3), Max(sum2, sum4)];
            if (tmp_time < total_time) {
                if (tmp_time = Max(sum1, sum3)) Task_seq = 0;
                    else Task_seq = 1;
                    Boundary_index ++; }
            else break; }
Step 3: Determine the task distribution
       if (Task_seq = 0)
         Tasks whose index are under the Boundary_index in deviceCPU
         and over the Boundary_index in deviceGPU run simultaneously
       else
         Same as the above except the sequence of workers
```

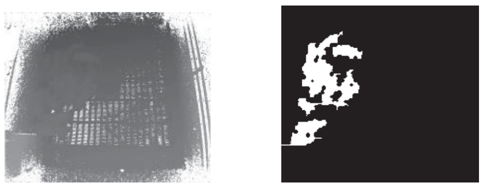
Fig. 4: Pipeline scheduling algorithm.

4 Experimental results

4.1 Thermal Comfort Assessment System

The camera was located 4 m above the floor to monitor a pig room that measured 4 × 3 m^2, and there were 13 weaning pigs 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) in order to handle the noise and unreliable values. From the images of size 256 × 212, we masked out some regions where pigs cannot be detected. A background image was computed as pixel-by-pixel average of the 10-minute video sequence without pigs.

Figure 5 (a) shows one example of the input image (i.e., Kinect depth image), and Figure 5 (b) shows the output image of the thermal comfort assessment system. First, we used spatiotemporal interpolation to reduce the noise and unreliable values. Then, we were able to extract the foreground by using background subtraction. The remaining noise could be eliminated by using morphology operators, and the area of touching-pigs was computed for assessing the thermal comfort automatically.



(a) Input image (b) Output image

Fig. 5: The result of the thermal comfort assessment system.

4.2 Speedup on CPU-GPU Heterogeneous Computing Platforms

To evaluate the proposed method, we conducted our experiment using two platforms. The first platform (denoted as "platform 1") consisted of a 3.50GHz Intel ® Core™ i5-4690 CPU with 4 cores, a GeForce GTX 760, and 8GB of RAM, whereas the second platform (denoted as "platform 2") consisted of a 2.67GHz Intel ® Core™ i5 CPU 750 with 4 cores, a GeForce GTS 250, and 8GB of RAM. That is, we considered platform 1 and platform 2 as a higher-performance GPU platform and a higher-performance CPU platform, respectively.

By measuring the communication time and computation time of each task on each platform, we could get the optimal solution, regardless of the relative performance between CPU and GPU of a given platform. As shown in Fig. 6, the speedup of the proposed method was better than that of GPU-only method (i.e. all tasks were executed on deviceGPU), regardless of platform 1 or platform 2. 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 manycore GPU is much larger than that of multicore CPU, the computational characteristics of task 4 does not match well with deviceGPU and thus the proposed method can provide superior performance than the GPU-only method (i.e. all tasks including task 4 were executed on deviceGPU). Based on the pipelined scheduling over multicore CPU and manycore GPU, the proposed method could obtain the performance of more than 200 fps on both platforms. Note that pig detection is the initial step for intelligent pig monitoring such as aggressive behavior monitoring [11], and thus faster pig detection is required for the final goal of real-time intelligent pig monitoring.

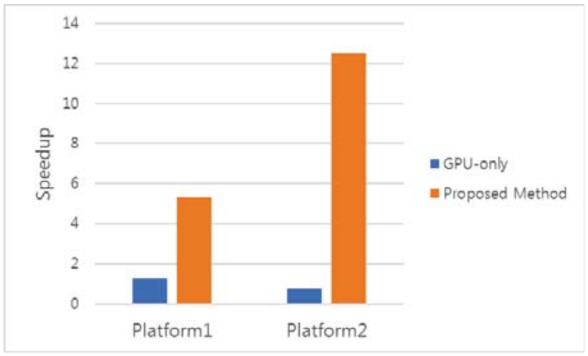


Fig. 6: Performance comparison of each platform

5 Conclusions

By using the depth information obtained from a Kinect sensor, we proposed a method to automatically detect pigs at night on a CPU-GPU heterogeneous computing platform. We reduced the noise in the depth information obtained from a

Kinect and then detected pigs with background subtraction. To satisfy the real-time requirement, we parallelized the whole monitoring system by determining a computing device for each task and balancing the execution times of CPU and GPU simultaneously. As a future work, using both data and task parallelism will be considered to reduce the total execution time further.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the MEST (2015R1D1A1A09060594).

References

[1] T. Banhazi, H. Lehr, J. Black, H. Crabtree, P. Schofield, M. Tscharke, and D. Berckmans, "Precision Livestock Farming: An International Review of Scientific and Commercial Aspects"; *International Journal of Agriculture and Biology*, Vol. 5, No. 3, pp. 1-9, 2012.

[2] A. Nasirahmadi, U. Richter, O. Hensel, S. Edwards, and B. Sturm, "Using Machine Vision for Investigation of Changes in Pig Group Lying Patterns"; *Computer and Electronics in Agriculture*, Vol. 119, pp. 184-190, Nov 2015.

[3] Y. Guo, W. Zhu, P. Jiao, and J. Chen, "Foreground Detection of Group-Housed Pigs based on the Combination of Mixture of Gaussians using Prediction Mechansim and Threshold Segmentation"; *Biosystems Engineering*, Vol. 125, pp. 98-104, Sep 2014.

[4] H. Kashiha, C. Bahr, S. Ott, C. Moons, T. Niewold, F. Odberg, and D. Berckmans, "Automatic Weight Estimation of Individual Pigs using Image Analysis"; *Computers and Electronics in Agriculture*, Vol. 107, pp. 38-44, Sep 2014.

[5] A. Wongsriworaphon, B. Arnonkijpanich, and S. Pathumnakul, "An Approach based on Digital Image Analysis to Estimate the Live Weights of Pigs in Farm Environments"; *Computers and Electronics in Agriculture*, Vol. 115, pp. 26-33, Jul 2015.

[6] P. Wouters, R. Geers, G. Parduyns, K. Goossens, B. Truyen, V. Goedseels, and E. Van der Stuyft, "Image-analysis parameters as inputs for automatic environmental temperature control in piglet houses"; *Computers and Electronics in Agriculture*, Vol. 5, No. 3, pp. 233-246, Dec 1990.

[7] B. Shao, and H. Xin, "A real-time computer vision assessment and control of thermal comfort for group-housed pigs"; *Computers and Electronics in Agriculture*, Vol. 62, No. 1, pp. 15-21, Jun 2008.

[8] J. Stone, D. Gohara, and G. Shi, "OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems"; *Computing in Science and Engineering*, Vol. 12, No. 3, pp. 66-73, 2010.

[9] B. Gaster, L. Howes, D. Kaeli, P. Mistry, and D. Schaa, "Heterogeneous Computing with OpenCL: Revised OpenCL 1.2 Edition"; *Newnes*, 2012.

[10] I. Jung, and C. Jeong, "Parallel Connected-Component Labeling Algorithm for GPGPU Applications"; *Proc. of ISCIT*, pp. 1149-1153, 2010

[11] J. Lee, L. Jin, D. Park, and Y. Chung, "Automatic Recognition of Aggressive Pig Behaviors using Kinect Depth Sensor"; *Sensors*, Vol. 16, No. 5, pp. 531, May 2016.