Parallel Implementations of Digital Focus Indices Based on Minimax Search Using Multi-Core Processors

HyungTae Kim[1*], Duk-Yeon Lee, Dongwoon Choi, Jaehyeon Kang and Dong-Wook Lee[2]
[1] Digital Transformation R&D Department, Korea Institute of Industrial Technology
Ansan, Gyeonggi, 15588, Korea
[e-mail: htkim@kitech.re.kr]
[2] Applied Robot R&D Department, Korea Institute of Industrial Technology
Ansan, Gyeonggi, 15588, Korea
[e-mail: kitech.re.kr]
[*]Corresponding author: HyungTae Kim

Received August 3, 2022; revised September 13, 2022; revised November 4, 2022; accepted December 4, 2022; published February 28, 2023

Abstract

A digital focus index (DFI) is a value used to determine image focus in scientific apparatus and smart devices. Automatic focus (AF) is an iterative and time-consuming procedure; however, its processing time can be reduced using a general processing unit (GPU) and a multi-core processor (MCP). In this study, parallel architectures of a minimax search algorithm (MSA) are applied to two DFIs: range algorithm (RA) and image contrast (CT). The DFIs are based on a histogram; however, the parallel computation of the histogram is conventionally inefficient because of the bank conflict in shared memory. The parallel architectures of RA and CT are constructed using parallel reduction for MSA, which is performed through parallel relative rating of the image pixel pairs and halved the rating in every step. The array size is then decreased to one, and the minimax is determined at the final reduction. Kernels for the architectures are constructed using open source software to make it relatively platform independent. The kernels are tested in a hexa-core PC and an embedded device using Lenna images of various sizes based on the resolutions of industrial cameras. The performance of the kernels for the DFIs was investigated in terms of processing speed and computational acceleration; the maximum acceleration was 32.6× in the best case and the MCP exhibited a higher performance.

Keywords: Digital Focus Index, Computer Vision Systems, Embedded Systems, Mobile Computing, Parallel Processing,

A preliminary version of this paper was presented at APIC-IST 2022, and was selected as an outstanding paper.

http://doi.org/10.3837/tiis.2023.02.014 **ISSN : 1976-7277**

1. Introduction

Multiple image sensors are mounted on various mobile devices and automatic focus (AF) has become an essential function for maximizing camera usability. The AF determines the focal position from the variation in the digital focus index (DFI) during actuating camera optics. DFI originated from the telescope in astronomy [1] and the microscope in bioengineering [2]; the DFI is derived from optical transfer functions and used to evaluate high-frequency elements in an image [3]. The variation in the DFI during AF forms a continuous and smooth curve with a single local maximum. Thus, the AF determines the focal position at the maxima of the DFI curve while actuating the camera optics [4]. The application of DFI is not limited to AF, and it can also be applied to various areas in machine vision, such as image fusion, feature extraction and stitching [5][6].

Further, AF is time-consuming because of the massive number of computational operations on the DFI. Hundreds of mega-pixel images are acquired and the DFI is sequentially computed during AF. Conventional DFI is equated as the sum of many iterative mathematical operations for all pixels [7]. Commercial devices reduce the computational requirement of the AF by limiting the focal area to a region-of-interest (ROI), rather than by accelerating the DFI equations. However, only a few studies on the computational acceleration of DFI have been conducted. Jin devised a hardware accelerator linking a camera with a framegrabber. The hardware accelerator was constructed using a field-programmable gate-array (FPGA) which lowered the computational load on the PC [8] but the hardware cost was relatively high because of various connection interfaces. Thus, the hardware approach is inconvenient for supporting the diverse formulations of DFIs. The software approach uses a general processing unit (GPU) platform to parallelize the DFI computation, and computed unified device architecture (CUDA) is applied to accelerate the Tenenbaum gradient [9]. The GPU platform is available on a PC or an embedded device. Another parallel device is the multi-core processor (MCP), which has stronger but fewer cores than a GPU. All these parallel devices are effective for lowering the computational cost of image processing, and therefore, they have been introduced in machine vision, such as in stereo vision [10], image compression [11], and video analysis [12]. However, the parallel architecture for DFI is rarely discussed even though its processing cost is high [9][13].

Range algorithm (RA) and image contrast (CT) are conventional DFIs. RA and CT in the current microcopy are based on serial programming. However, the processing time becomes crucial for automatic focus as a resolution of a commercial camera reaches dozens of mega bytes. Thus, parallel architectures of RA and CT were proposed in this paper for reducing the processing time of AF. A parallel architecture was designed to accelerate the minimax search in RA and CT, and parallel kernels were constructed for a GPU and MCP using cross-platform open-source softwares. The performance of the kernels was verified using various image sizes of industrial cameras. The rest of this paper is organized as follows: Section 2 reviews RA and CT, Section 3 describes the parallel architecture and Section 4 presents the acceleration performance of the experiments. Conclusions are presented in Section 5.

2. Background

DFIs are usually derived from the gradient between image pixels because pixel changes occur in high frequency content and object edges [3]. The general focus function is defined as the

following equation.

$$F_{nm\theta} = \iint E\left[\left|\frac{\partial^n I(x,y)}{\partial x^n}\right| - \theta\right]^m dxdy \quad (1)$$

where F, I, θ and E represent the DFI, gray level in a digital image at (x,y), threshold, and rectified linear function, respectively. The focus position z_f is determined at the maximum of the DFI curve after moving the working distance between the camera and the object z. Images are acquired, and the DFIs are calculated at every small increment Δz.

$$z_f = \max_z F(I_z) \quad (2)$$

In addition to the gradient methods, DFIs are obtained using statistical, intuitive and histogram approaches. In this study, the RA and CT are obtained from the minimum and maximum gray levels in the histogram. RA in the previous study is applied to fluorescence microscopy [14] and 3D reconstruction [15], and it is evaluated from the difference between the minimum and maximum gray levels [16].

$$F_{RA} = \max(k|H_k > 0) - \min(k|H_k > 0) \quad (3)$$

where, k and H represent the gray level and histogram of I, respectively. A defocused image contains a single gray shade, whereas an in-focus image contains diverse gray levels. Therefore RA increases when an image plane approaches a focal position.
The CT originated from the Michelson contrast, which evaluates the gray-level diversity in a focused image [17][18]. CT is based on periodic optical responses derived from the Michelson formula, and it is measured from the minimum and maximum luminance as [19].

$$F_{CT} = \frac{L_{max} - L_{min}}{L_{max} + L_{min}} \quad (4)$$

where L represents luminance. The concepts of these methods were utilized to maximize the contrast of target images in bio-medical imaging. The contrast ratio of the human skin tissue is measured from the gray levels of the vein and skin areas [20]. The cell images of the dark field, bright field, and phase contrast are obtained using a concept similar to equation (4) [21][22]. The minimum and maximum luminance values in an image are equal to the minimum and maximum gray levels in the histogram obtained using equation (3). Thus, RA and CT originated from different theories but can share the processing values of a histogram by finding the minimum and maximum gray levels. The processing costs of RA and CT are lower than those of other DFIs; therefore RA and CT are used to verify the focusing performance of the microscope. Previous studies focused on the verification of the focusing theory and accuracy and the processing cost was not considered.

3. Parallel Implementation

A GPU and an MCP are available for parallel processing; DFIs, however, are conventionally developed using serial processing conditions. As the camera resolution increases, serial processing exposes the limit of processing time. Thus, parallel processing using a GPU and an MPC is one of the solutions to lower the processing cost of the DFI. The sequence of parallel processing for the DFI is shown in **Fig. 1**. A single-core assesses data and computes a focus

function on each pixel in the serial processing case. In case of the parallel processing, the image data is partitioned to subsets and the serial processing is allocated to each of the processors. The DFI is then obtained by gathering the results of the subsets. The parallel processing of the DFI involves focus function and reduction. The focus function evaluates the degree of focus in each pixel, whereas reduction gathers and summarizes the evaluated results of the full set of pixels.

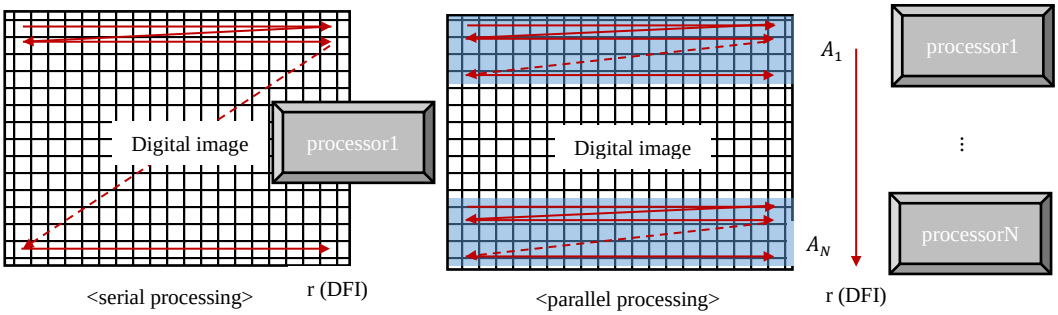

Fig. 1. Processing sequences of DFI in serial and parallel processing.

3.1 Parallel reduction

RA and CT commonly search for the minimum and maximum gray levels in an image. A parallel search for minimax can accelerate the computation of RA and CT, and a parallel search for minimax can be implemented using parallel reduction. Parallel reduction is a fundamental architecture for obtaining the sum and basic arithmetic from large-scale data [23]. Further, image $I(x,y)$ is a large dataset, and the image can be linearly mapped into a 1D array, $A(p)$. The array size amounts to millions but because the processing cores of the GPU and MCP are limited, the array should be partitioned according to the number of parallel cores N. A_i, the subsets of A, can be organized from various combinations based on the reduction structures; the following conditions must be satisfied:

$$\cup_{i=1}^{n} A_i = A \quad A_i \neq \emptyset \quad A_i \cap A_j = \emptyset \; (i \neq j) \tag{5}$$

Parallel reduction obtains a scalar result from the set, array size and arbitrary arithmetic operator, $\odot$ [24].

$$r = reduce(A, N, \odot) \tag{6}$$

The parallel reduction in Equation (6) can be further formulated by associating the subsets as [25].

$$r = A_1 \odot \cdots \odot A_N = \coprod_{i=1}^{N} \odot A_i \tag{7}$$

In the minimax search problem, the binary operator, $\odot$, is substituted with comparison operators, such as < and >. The RA and CT values are calculated from r_{min} and r_{max} after reduction.

3.2 Parallel architecture

The parallel architecture for RA and CT was constructed to search for both the minimum and maximum gray level in an image, as shown in **Fig. 2**. The parallel architecture comprises preliminary and reduction steps. The preliminary step involves performing minimax search in subsets; the reduction step determines the minimum and maximum gray level of an image from the subsets. They are partitioned by allocating data addresses linearly to the parallel cores when the pixel data in an image were loaded onto parallel memory. Normally, all pixel data in an image are converted into decimal data. When a mask image is additionally applied to limit the processing area of an image, the image data on the active pixels of the mask are converted. The open sources for machine vision provide multiple data formats, and therefore, the parallel architecture is implemented using a template variable for a variable data format such as unsigned char, short, float, and double.

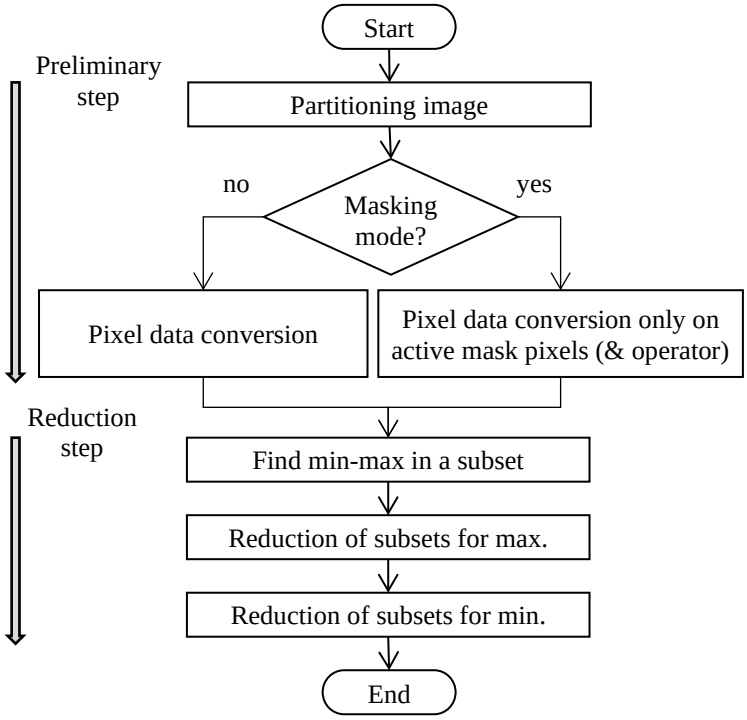

Fig. 2. Processing flow in the parallel architecture of range algorithm and contrast.

After determining the minima and maxima in the subsets, the conventional parallel reduction was applied to search for the extreme values of an image [26]. The parallel reduction iteratively performs specific mathematical operations on pairs of elements in the array and transfers the result to the front element. Therefore, the results of the subsets are substituted into a parallel reduction to determine the minima and maxima from pairs of subsets. The eliminated elements of pairs after the decision are discarded, and the surviving elements are transferred to the front elements. The widths of the surviving elements are halved in every iteration. The iteration was terminated when the width was 1 and the result was placed at the head element of the array. **Fig. 3** shows the concept of parallel reduction in a parallel architecture. The band of threads is much smaller than the number of pixels, thus the image is sliced into subsets by the band width and then the parallel reduction is iteratively performed on the subsets. The parallel reduction is applied to the local minima and maxima of the subsets to

find the global minimum and maximum of the image. The RA and CT were then calculated using the global minimum and maximum values.

4. Experiments

Previous studies implemented algorithms on one parallel platform such as CUDA or OpenMP [27][28]. In this study six parallel platforms were constructed using C++, OpenCV, CUDA, OpenCL, OpenMP, and TBB for RA and CT. The C++ subroutine is based on serial processing using a single CPU task. Further, the OpenCV subroutine is implemented using the cv::minmaxLoc function and used as a reference for verification. CUDA and OpenCL were applied to GPU subroutines, and OpenMP and TBB were employed for the MCP subroutines. The parallel architecture in **Fig. 3** was adopted to the GPU subroutines. The number of MCP cores was much smaller than that of the GPU cores and the MCP subroutines were constructed using serial programming. The subroutines were integrated into library functions, whose I/O parameters and naming were decided based on the OpenCV platform. The subroutines automatically detected image properties and determined the data type based on pixel depth: unsigned char (8 bit), unsigned short (16 bit), signed short (16 bit), float (32 bit) and double (64 bit). The template variable is applied to the subroutines to share the algorithm codes with an arbitrary data type. Many functions in the machine vision library normally perform mathematical operations on the entire image and provide ROI and masking operations for a sub-image and on the active pixels of the mask. Further, the subroutines for RA and CT were designed to support the ROI and masking operations.

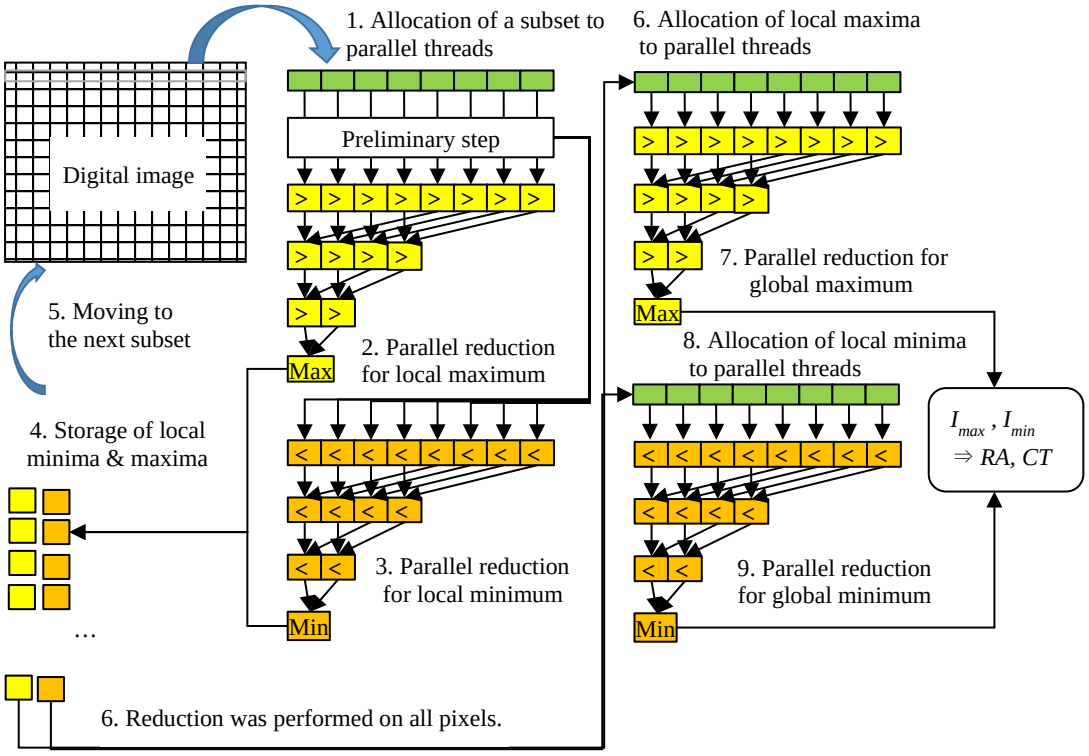

Fig. 3. Parallel architecture using reduction to find min-max for range algorithm and contrast.

Sample images in the experiments were synthesized by blurring, arranging, and tiling the Lena images [29]. Simulated AF is conducted to verify the DFI by blurring the Lena images using an average smoothing filter [30] with a linear size enlargement, as shown in **Fig. 4**. For AF, defocused images are blurred, but focused images are distinct; thus, the blur intensity is related to the virtual working distance (VWD) between a virtual camera and an object. The VWD of the original Lena image can be focused on, and that of the other images is defocused.

Fig. 4. Original Lena and smoothed images (filter size=50) for simulated autofocus.

The size of the sample images ranged from 0.25 to 45 mega pixels according to the resolution of the industrial cameras. The pixel depth of the sample images was varied from 8 bit to 64 bit based on the data types. In the experiment, the gray level of each image was increased at an offset of 0 – 100 × bytes. The processing times for the RA and CT were measured in 100 repetitions at each offset; the tests were performed on the ROI and masking operations. The target and active regions of the synthesized images were defined in the bottom-right tile of the Lena image. Thus, six subroutines were tested using 56 sample images for five data types, and the processing time was measured after 10,000 repetitions using each sample image. Further, the subroutines were tested for normal, ROI, and masking operations. These benchmark tests were conducted on a hexa-core PC (AMD Ryzen 3950X , RTX 2070, 64GB, Ubuntu 20.04), an embedded device (Jetson AGX Xavier, 32 GB, Ubuntu 18.04) and a stand-alone machine vision (Crevis IPC Vision Block, i7-6600U, 4GB, Ubuntu 22.04). OpenCL was supported for the hexa-core PC and the stand-alone machine vision, but was incompatible with the embedded device. CUDA was supported for the hexa-core PC and the embedded device.

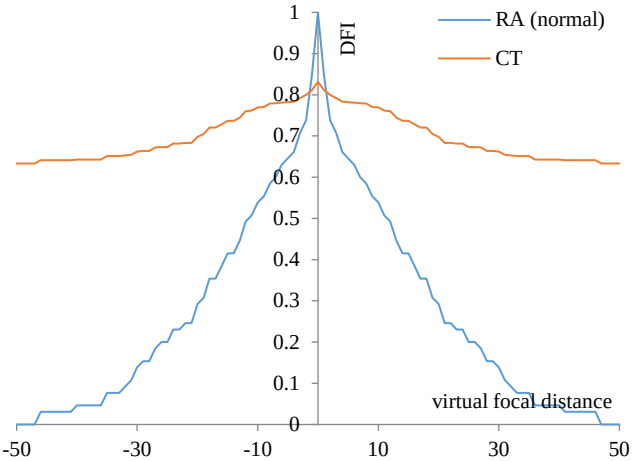

Fig. 5. Variation of range algorithm and contrast in simulated autofocus.

5. Results and Discussions

The subroutines for AF were verified using the simulated AF. The normalized RA was 0 when the VWD was far from the focus. Further, the RA reached the maximum value at the virtual focal position; the CT showed the offsets when defocused, but it exactly indicated the maximum at the virtual focal position. Therefore, the subroutines can be used for measuring the DFI applicable to the AF. The variation in the simulated AF is shown in **Fig. 5**. The benchmark results for the RA resemble those of CT, and the tendency is similar. The following results are discussed for the sake of the RA.

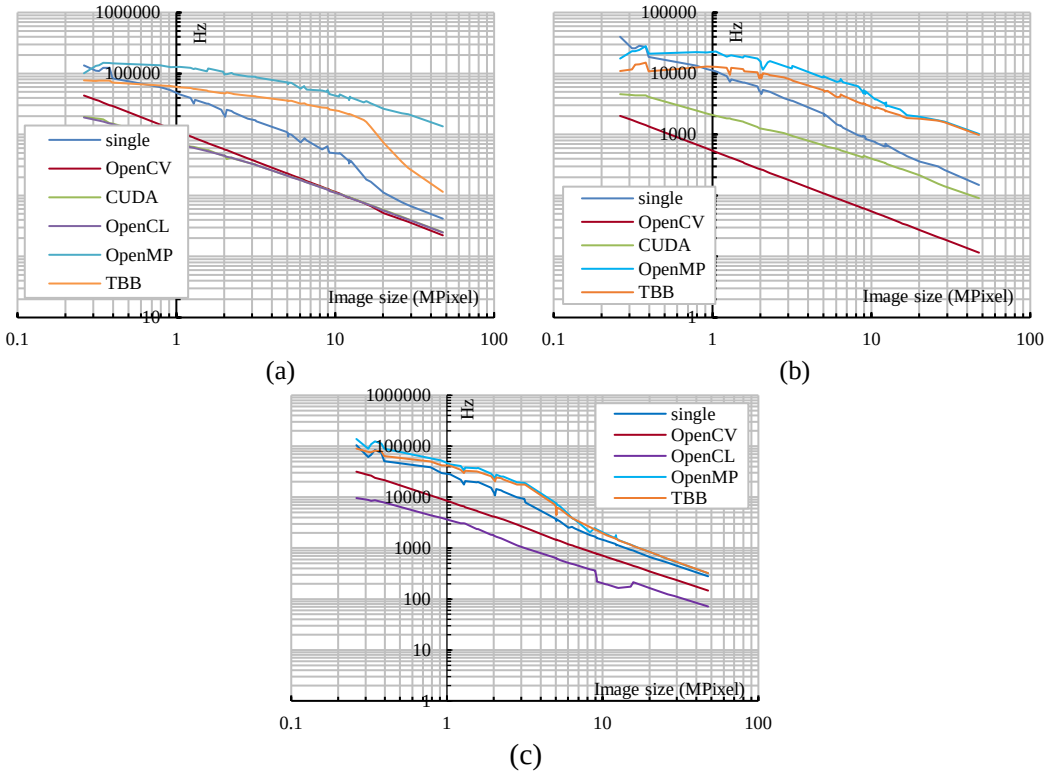

Fig. 6. Variations of the processing speed using (a) a hexa-core PC, (b) an embedded device, and (c) a stand-alone machine vision in the 8-bit normal operation.

5.1 Results for normal operations

Fig. 6 shows the variations in processing speed using the hexa-core PC, embedded device, and the stand-alone machine vision in the case of the 8-bit images under normal operation. The horizontal axis is the image size in mega-pixels: the vertical axis is the frame rate per second. Both axes are log-scaled. The processing speed decreased based on the image size and the rate of decrease was approximately linear. The processing time of the hexa-core was 200 – 150,000 Hz. Therefore, the real-time processing was performed using subroutines. **Fig. 7** shows acceleration performances using the hexa-core PC, embedded device, and stand-alone machine vision according to the pixel depth. The pixel depth was expressed with the number of data bits and data types.

The reference of acceleration was single-core processing using C++. The I-shaped vertical lines on each bar indicate the minimum and maximum accelerations. For the hexa-core PC,

OpenMP was superior to the other platforms, followed by TBB. The processing of the MCP was faster than that of the GPU, and the acceleration increased significantly with an increase in image size. The maximum acceleration was 32.6× in an 8-bit pixel depth, but the average maximum had a 16-bit pixel depth. Further, the MCP was overwhelmed in the experiments with the embedded device. The average maximum occurred using OpenMP and TBB at 32-bit and 64-bit pixel depth, respectively. The maximum acceleration was 7.7× when using TBB at a 32-bit pixel depth. The MCP processing also showed a better acceleration in the case of the stand-alone machine vision. The parallel processing using the MCP was faster than those using other types of processing in terms of normal operations.

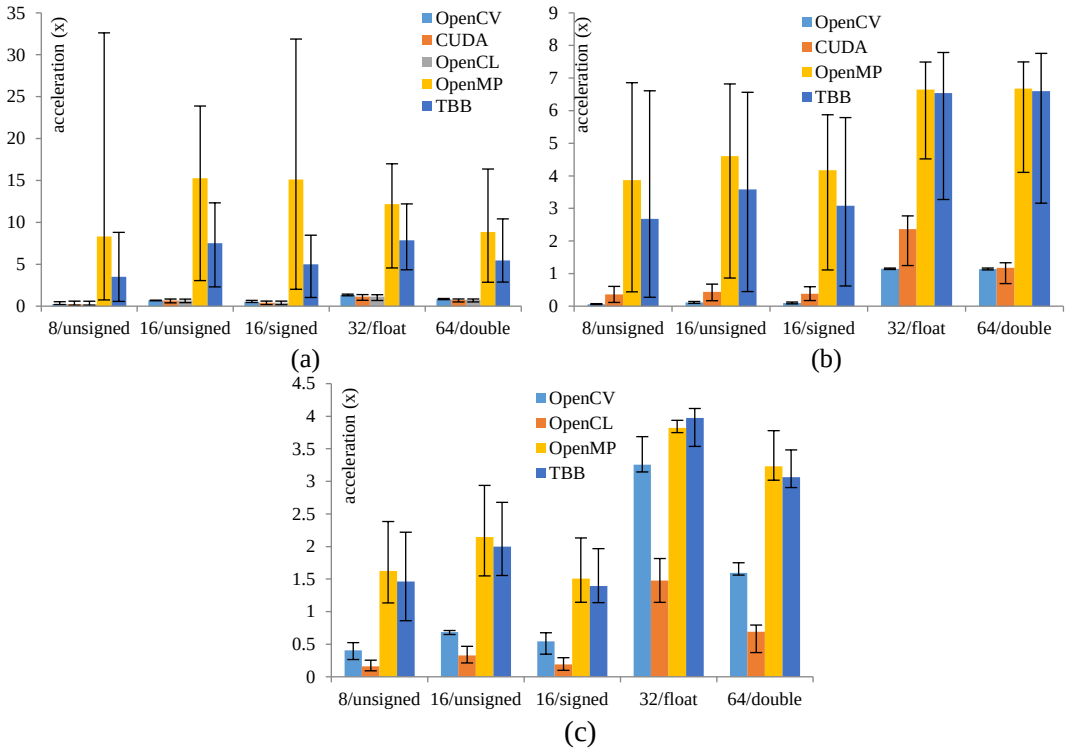

Fig. 7. Acceleration of normal operation using (a) a hexa-core PC, (b) an embedded device, and (c) a stand-alone machine vision according to pixel depth.

5.2 Results for ROI operations

Fig. 8 shows the variation in processing speed using the hexa-core PC, embedded device, and stand-alone machine vision in the case of the 16-bit images under the ROI operation. This case indicates that the processing time was approximately constant based on the image size. The ROI was performed on a specific area of the sample images; however, the processing load remained unchanged. The processing times of the hexa-core PC, embedded device and stand-alone machine vision were approximately 12,000-130,000, 1,800 – 20,000 and 2,200 – 60,000 Hz, respectively; thus, real-time processing is possible. **Fig. 9** shows the acceleration performances using the hexa-core PC, embedded device, and stand-alone machine vision based on the pixel depth.

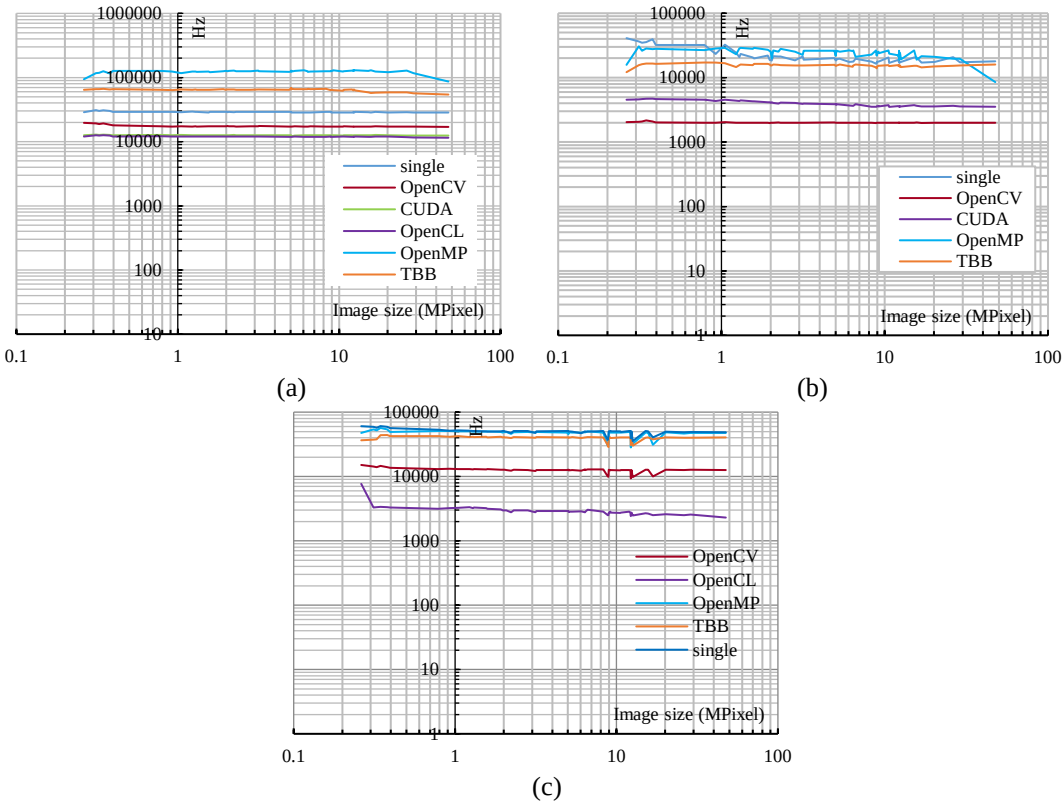

Fig. 8. Variations of the processing speed using (a) a hexa-core PC, (b) an embedded device, and (c) a stand-alone machine vision in the 16-bit ROI operation.

OpenMP was superior to the other platforms using the hexa-core; TBB was second, and the GPU was slower in this case. However, the single-core processing showed similar performances to the MCP in the embedded device, and stand-alone machine vision; this implies that initial time is required to activate the MCP. The numbers of cores were fewer than that of the hexa-core, and the data size of the ROI operation was much smaller than that of the normal operation. The acceleration was relatively constant though the image size increases; however, the deviation of the acceleration is lower than that of normal operation. The maximum acceleration using the hexa-core was 9.3× for a 64-bit pixel depth, but the average maximum was 7.8× in the hexa-core PC. The maximum accelerations using the embedded device and stand-alone machine vision were 5.6× and 3.9× respectively. The MCP generally exhibited higher performance than that of the others. The average maximum occurred in the cases of OpenMP and TBB for the 32-bit and 64-bit pixel depths, respectively. Further, this also indicates that parallel architecture is effective for large-scale data processing.

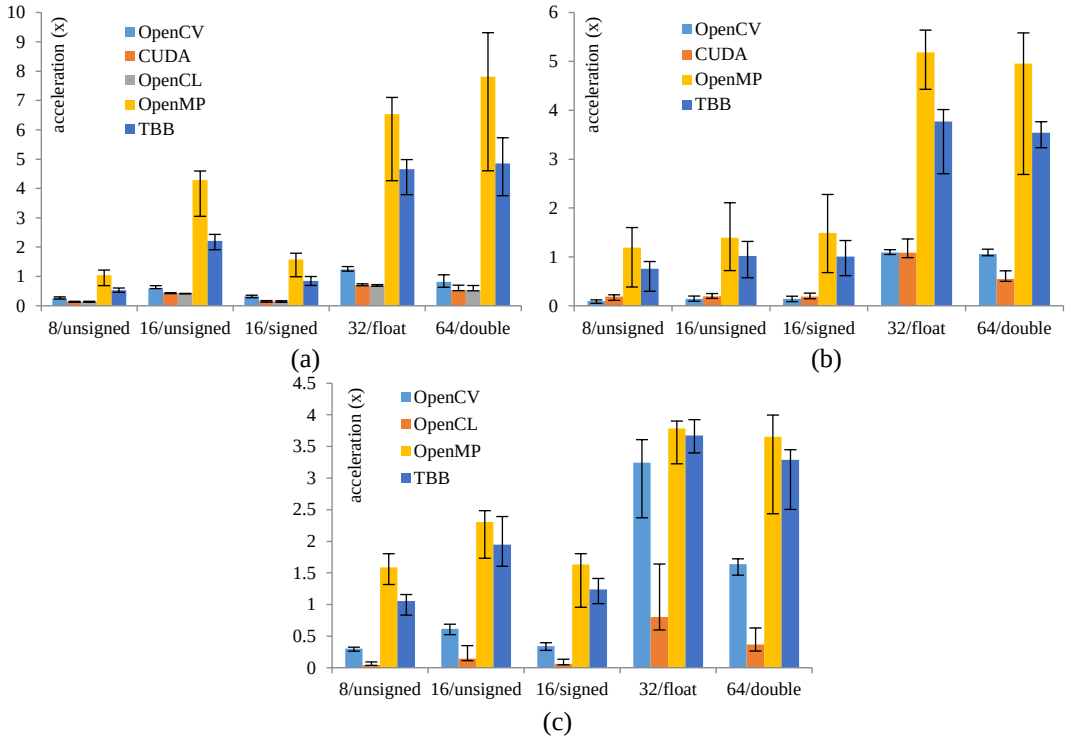

Fig. 9. Acceleration of ROI operation using (a) a hexa-core PC, (b) an embedded device, and (c) a stand-alone machine vision according to pixel depth.

5.3 Results for masking operations

Fig. 10 shows the variation in the processing speed using the hexa-core PC, embedded device, and stand-alone machine vision for the 64-bit images under the masking operation. The processing time decreased linearly with image size. In this case the results were classified into two groups, vis. MCP, and the others. OpenMP and TBB are available for real-time processing because other platforms exhibit low performances for large image sizes. **Fig. 11** shows the acceleration performances obtained using the hexa-core and embedded PCs according to the pixel depth.

OpenMP and TBB exhibited similar accelerations with large offsets. MCP was superior to the other platforms in the results of the 32-bit and 64-bit images. Single-core processing was better when using 8-bit and 16-bit images. This can be caused by the initial time required to activate the MCP. The maximum acceleration was 16.9× in the 64-bit pixel depth, and the average maximum was 13.2× in the hexa-core PC. The average maximum in the hexa-core PC and stand-alone machine vision occurred with OpenMP. TBB was superior in case of the embedded device. The average acceleration was 8.5× and 4.1× in the embedded device and stand-alone machine vision, respectively. Thus, parallel architectures for the MCPs were available in the acceleration for RA and CT in the case of the large data size.

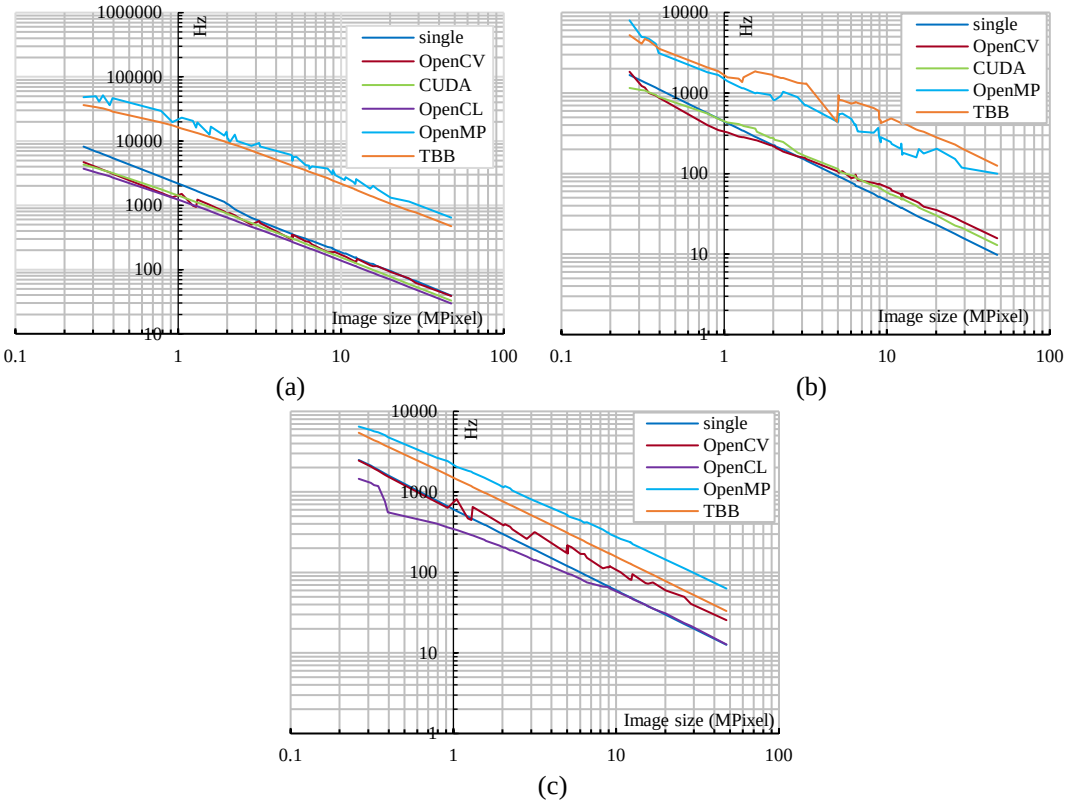

Fig. 10. Variations of the processing speed using (a) a hexa-core PC, (b) an embedded device, and (c) a stand-alone machine vision in the 64-bit masking operation.

5.4 Discussions

The performance of the hexa-core PC was better than that of the embedded PC because of the hardware specifications. The performance of the MCP was superior to those of the others in the hexa-core and embedded PC. The GPU acceleration was ineffective because of the chronic bottle neck in the image data transfer. The AF sequentially generates images and transfers the data of image pixels to a parallel device. Thus, the MCP is advantageous for these frequent data transfers, but the GPU is less favorable to compute the DFI. Therefore, an MCP is desirable for the computation of RA and CT; the acceleration is improved in larger images and higher data bits, and therefore, the MCP is more efficient for larger volumes of data. As shown in the ROI and masking operations, the MCP was effective for 32-bit and 64-bit images. The data quantities of these operations were much less than that of the normal operation. The initial time is necessary for starting the MCP. Thus, the MCP was suitable for a large-size and high-depth image. The embedded device and stand-alone machine vision had octa-core and dual-core processors targeting mobile devices. The performance of these processors is much lower than that of the hexa-core PC. Nonetheless, the MCP was effective in most cases. Thus, it is a worthwhile attempt to apply the MCP for the DFI acceleration.

The performance is competitive with the reference of previous studies. Although the DFIs tested in previous studies were different from those in this study, the processing speed can be used as a reference value. A hardware accelerator using a Sparatan-3 FPGA processor accomplished 140 Hz for a 640 × 480 image [8]. Comparing performances of a recent UltraScale+ FPGA, the transceiver speed of the recent FPGA was increased to 93.2 times.

Thus, the maximum processing speed achieved using a recent FPGA is estimated at 13,054 Hz. The processing speed of OpenMP for the same size ranged from 22,000 to 148,000 Hz, as shown in **Fig. 6**. A software accelerator using Jetson TX1 achieved 120–250 Hz for a 1.2 MP image, according to the DFIs [9]. Considering the performances of recent Jetson AGX Xavier, the processing speed is estimated at 3,840–8,000 Hz without considering the complexity of the DFIs in the software accelerator. The processing speed in this study was 18,916 Hz, as shown in Fig. 6(b). Recording speeds of recent smart phones are 60 – 120 Hz for 4K full HD video, and acquisition speeds of machine vision cameras are usually 120 – 220 Hz. The result in Fig. 6(b), using the ARM processor, is estimated at 2,584 Hz for the 4K full HD image, thus the proposed architecture is usable with mobile and industrial machine vision devices.

The C++ functions of RA and CT were implemented in the SDK; therefore, these DFIs are applicable to industrial applications and mobile devices. Parallel reduction is a common process in various image operations and DFIs. Therefore, the kernels can be reused to develop algorithms. In contrast to minimax in the RA and CT, the DFIs generally comprise arithmetic operations and the sum operand. Thus, other DFIs can be parallelized by modifying other arithmetic operators.

In addition, their performances of related studies using 8-bit, fixed-size and their own images were shown. Our study experimented with various processors, variable bit-depth, commercial sizes, library functions and Lena image for general conditions.

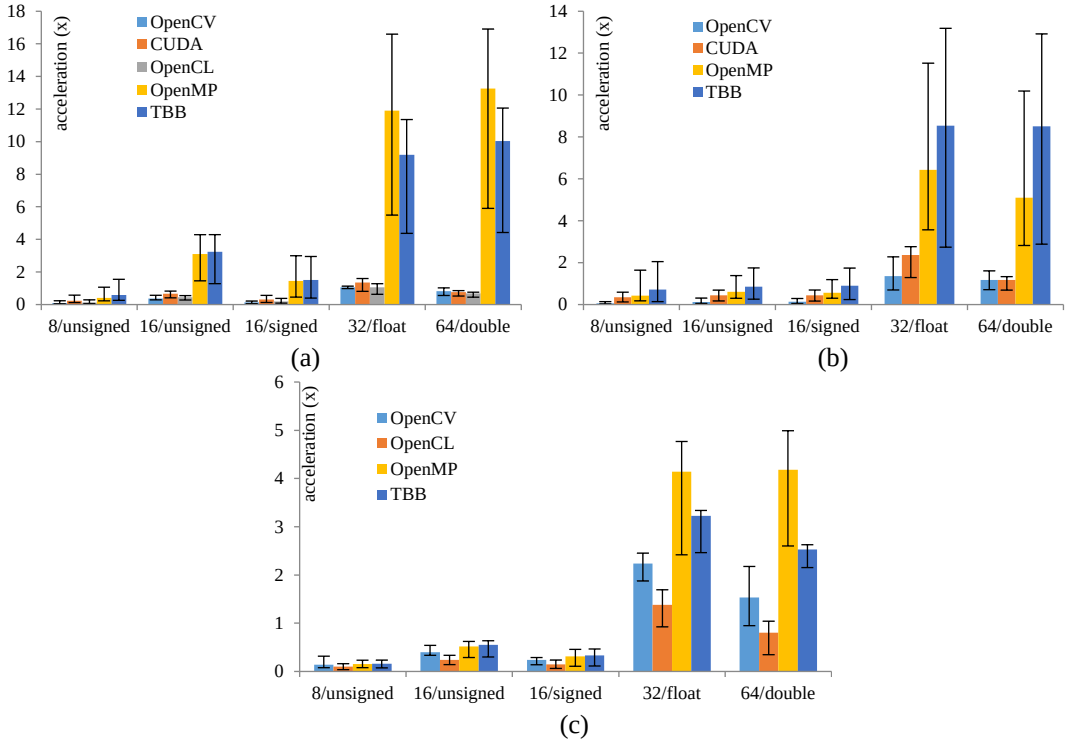

Fig. 11. Acceleration of the masking operation using (a) a hexa-core PC, (b) an embedded device, and (c) a stand-alone machine vision according to pixel depth.

6. Conclusions

Parallel architectures of RA and CT were proposed to accelerate AF and were constructed using open sources. These architectures were based on searching for the minimum and maximum gray levels in an image; they comprise preliminary and reduction steps. The proposed architectures can be designed using conventional parallel reduction. The kernel was constructed using the GPU and an MCP considering cross-platform open-source softwares. The kernel was designed to perform normal, ROI, and masking operations, and the kernel can handle arbitrary data types using a template variable. In the experiment, divese types of images were tested to measure the processing time using a high-performance desktop, an embedded device and a stand-alone machine vision. The performance of the MCP was superior to that of the GPU as well as conventional processing. The MCP was more effective on a large-size and high-depth image. Thus, the MCP is advantageous in processing large image data. Although the MCP was ineffective in processing small-image data, the MCP showed a great acceleration in most cases. Thus, it is necessary to attempt applying the MCP to accelerate AF. The results also imply that the proposed architecture can improve the computational performance of AF in industrial machines and mobile devices. Because of restrictive conditions in mobile devices, the specifications of a mobile CPU are lower than those of a desktop. AF is an indispensable function in many mobile devices, and with ever-increasing camera pixels, the MCP can be viable alternative for DFI acceleration using the restrictive processing resources in mobile devices. In addition, we are currently testing parallel architectures of other DFIs proposed in previous research and will present the results soon.

Acknowledgement

We would like to acknowledge the financial support from the Year 2022 Culture Technology R&D Program of Ministry of Culture, Sports and Tourism and Korea Creative Content Agency (Project Name: Development of the system for digital data acquisition of modern and contemporary fine arts and supporting science-based art credibility analysis, Project Number:R2020060004).

References

[1] R. A. Muller and A. Buffington, “Real-time correction of atmospherically degraded telescope images through image sharpening,” *J. Opt. Soc. Ame.*, vol. 64, pp. 1200-1210, Jan. 1974. Article (CrossRef Link)

[2] M. L. Mendelsohn, D. E. Bennett, E. Bogart, and B. H. Mayall, “Computer-Oriented Analysis of Human Chromosomes IV. Deoxyribonucleic Acid-Based Centromeric Index,” *J. of Histochem. and Cyto.*, vol. 22, no. 7, pp. 554-560, Jul. 1974. Article (CrossRef Link)

[3] F. C. Groen, I. T. Young and G. Ligthart, “A comparison of different focus functions for use in autofocus algorithms,” *Cytometry*, vol. 6, no. 2, pp. 81-91, Mar. 1985. Article (CrossRef Link)

[4] A. R. Cabazos-Mar´ın and J. A´ lvarez-Borrego, “Automatic focus and fusion image algorithm using nonlinear correlation: Image quality evaluation,” *Optik*, vol. 164, pp. 224-242, Jul. 2018. Article (CrossRef Link)

[5] Y. Yang, W. Zheng and S. Haung, “A Novel Automatic Block-based Multi-focus Image Fusion via Genetic Algorithm,” *KSII Trans. Inter. Info. Sys.*, vol. 7, no. 7, pp.1671-1689, Jul. 2013. Article (CrossRef Link)

[6] R. Jia, A. Nahli, D. Li and J. Zhang, “A Multi-Freature Extraction-Baed Algorithm for Stitching Tampered/Untampered Image Classification,” *App. Sci.*, vol. 12, no. 5, pp. 2337, Feb. 2022. Article (CrossRef Link)

[7] H. Lee, C. S. Jung and K. W. Kim, "Feature Preserving Autofocus Algorithm for Phase Error Correction of SAR Images," *Sensors*, vol. 21, no. 7, pp. 2370, Mar. 2021. Article (CrossRef Link)
[8] S. Jin, J. Cho, K. H. Kwon and J. W. Jeon, "A dedicated hardware architecture for real-time auto-focusing using an FPGA," *Mach. Vis. App.*, vol. 21, pp. 727–734, Feb. 2010. Article (CrossRef Link)
[9] J. M. Castillo-Secilla, M. Saval-Calvo, L. Medina-Vald`es, S. Cuenca-Asensi, A. Mart´ınez-A´ lvarez, C. Sa´nchez, and G. Cristo´bal, "Autofocus method for automated microscopy using embedded GPUs," *Biomed. Opt. Exp.*, vol. 8, pp. 1731-1740, Mar. 2017. Article (CrossRef Link)
[10] C. G. Kim and Y. S. Choi, "Improved Disparity Map Computation on Stereoscopic Streaming Video with Multi-core Parallel Implementation," *KSII Trans. Inter. Info. Sys.*, vol. 9, no. 2, pp.728 - 741, Feb. 2015. Article (CrossRef Link)
[11] C. Li, "Parallel Implementation of the Recursive Least Square for Hyperspectral Image Compression on GPUs," *KSII Trans. Inter. Info. Sys.*, vol. 11, no. 7, pp.3543 - 3557, Jul. 2017. Article (CrossRef Link)
[12] K. Liao, F. Zhao and M. Zhang, "Parallel Implementation Strate for Content Based Video Copy Detection Using a Multi-core Processor," *KSII Trans. Inter. Info. Sys.*, vol. 8, no. 10, pp.3520 - 3537, Oct. 2014. Article (CrossRef Link)
[13] H. Kim and D. W. Lee, "Parallel Processing Architecture of Intuitive Digital Image Indices Based on Open Sources," in *Proc. of IEEE TENSYMP*, Jeju, Korea, 2021.
[14] A. Santos, C. Ortiz De Sólorzano, J. J. Vaquero, J. M. Peña, N. Malpica and F. Del Pozo, "Evaluation of autofocus functions in molecular cytogenetic analysis," *J. Micro.*, vol. 188, no. 3, pp. 264-272, Oct. 1997. Article (CrossRef Link)
[15] S. Pertuz, D. Puig, and M. A. Garcia, "Analysis of focus measure operators for shape-from-focus," *Pat. Recog.*, vol. 46, pp. 1415-1432, May 2013. Article (CrossRef Link)
[16] L. Firestone, K. Cook, K. Culp, N. Talsania, and K. Preston Jr, "Comparison of Autofocus Methods for Automated Microscopy," *Cytometry*, vol. 12, no. 3, pp. 195-206, Mar. 1991. Article (CrossRef Link)
[17] E. Peli, "Contrast in complex images," *J. Opt. Soc. Ame. A*, vol. 7, no. 10, pp. 2032-2040, Oct. 1990. Article (CrossRef Link)
[18] G. Vivone, R. Restaino, M. D. Mura, G. Licciardi, and J. Chanussot, "Contrast and Error-Based Fusion Schemes for Multispectral Image Pansharpening," *IEEE Geo. Remo. Sens. Lett.*, vol. 11, no. 5, pp. 930-934, May 2014. Article (CrossRef Link)
[19] S. Yazdanfar, K. B. Kenny, K. Tasimi, A. D. Corwin, E. L. Dixon and R. J. Filkins, "Simple and robust image-based autofocusing for digital microscopy," *Opt. Exp.*, vol. 16, no. 12, pp. 8670-8677, Jul. 2008. Article (CrossRef Link)
[20] F. Wang, A. Behrooz, M. Morris, and A. Adibia, "High-contrast subcutaneous vein detection and localization using multispectral imaging," *J. Biomed. Opt.*, vol. 18, no. 5, pp. 050504, May 2013. Article (CrossRef Link)
[21] D. Jung, J. Choi, S. Kim, S. Ryu, W. Lee, J. Lee and C. Joo, "Smartphone-based multi-contrast microscope using color-multiplexed illumination," *Sci. Rep.*, vol. 7, pp. 7564, Aug. 2017. Article (CrossRef Link)
[22] Z. F. Phillips, M. V. D'Ambrosio, L. Tian et al, "Multi-Contrast Imaging and Digital Refocusing on a Mobile Microscope with a Domed LED Array," *PLoS ONE*, vol. 10, no. 5, pp. e0124938, May 2015. Article (CrossRef Link)
[23] S. Collange1, D. Defour, S. Graillat and R. Iakymchuk, "Numerical Reproducibility for the Parallel Reduction on Multi- and Many-Core Architectures," *Para. Comp.*, vol. 49, pp. 83-97, Nov. 2015. Article (CrossRef Link)
[24] B. K. Szymanski, W. Maniatty and B. Sinharoy, "Simultaneous Parallel Reduction on SIMD Machines," *Par. Proc. Lett.*, vol. 5, no. 3, pp. 437-449, Sept. 1995. Article (CrossRef Link)
[25] W. He, H. Guo, T. Peterka, S. Di, F. Cappello and H. Shen, "Parallel Partial Reduction for Large-Scale Data Analysis and Visualization," in *Proc. of IEEE LDAV*, Berlin, Germany, pp. 45-55, 2018.

[26] M. Harris, "Optimizing cuda - SC07: High Performance Computing With CUDA," *NVIDIA Developer Technology*, 2007.
[27] P. Borovska and M. Lazarova, "Efficiency of Parallel Minimax Algorithm for Game Tree Search," in *Proc. of Int. Conf. Comp. Sys. Tech.*, Rousse, Bulgaria, 2007.
[28] K. Rocki and R. Suda, "Parallel Minimax Tree Searching on GPU," in *Proc. of Int. Conf. Para. Proc. App. Math.: Part I*, Wroclaw, Poland, pp. 449–456, 2009.
[29] S. Ahmed, K. K. Ghosh, S. K. Bera, F. Schwenker and R. Sarkar, "Gray Level Image Contrast Enhancement Using Barnacles Mating Optimizer," *IEEE Acc.*, vol. 8, pp. 169196-169214, Sept. 2020. Article (CrossRef Link)
[30] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th Ed., New York, NY, USA: Pearson, 2018.
[31] Z. Jiang, Y. Kong, F. Liu, C. Liu and S. Wang, "Graphics processing unit (GPU) aided wavefront-based autofocusing in microscopy," *AIP Adv.*, vol. 8, pp. 105328, Oct. 2018. Article (CrossRef Link)
[32] J. C. Valdiviezo-N, F. J. Hernandez-Lopez, C. Toxqui-Quitl, "Parallel implementations to accelerate the autofocus process in microscopy applications," *J. Med. Imag.*, vol. 7, no. 1, pp. 014001, Jan. 2020. Article (CrossRef Link)

HyungTae Kim is working as a principal research engineer in Digital Transformation R&D Department in Korea Institute of Industrial Technology. He has authored and coauthored more than 50 articles in major journals and 160 articles in conference proceedings. His area of expertise is the machine vision algorithm interlocked with cameras, illumination, optics, kinematics, sensors and parallel devices. He recently achieved results from hyper-spectral, fluorescene, polarized and high-speed imaging.

Duk-Yeon Lee received the B.S degree in In- formation & Communication Engineering from Myong ji University, Young-in, Korea, and MS degrees in Computer Science Engineering from Hanyang University, Seoul, Korea, in 2004 and 2007 respectively. He is currently a senior researcher at Korea Institute of Industrial Technology (KITECH), Korea, since 2009. His research interests include Android Robots, Deep Learning and Image Processing.

Dongwoon Choi received the B.S., M.S. degrees in Mechanical Engineering from Hanyang University, Korea, in 2005, 2007, respectively. He is currently a principal researcher at Korea Institute of Industrial Technology(KITECH), Korea, since 2007. His research interests include mechanical design, android robot, humanoid robot, AI and smartfarm system.

Jaehyeon Kang received the B.S. and Ph.D. degrees in electrical engineering from Korea University, Seoul, South Korea, in 2011 and 2018, respectively. Since 2019, he has been a Senior Researcher with the Robotics Research and Development Department, Korea Institute of Industrial Technology, Ansan, South Korea. His research interests include sensor calibration, robot localization, environment mapping, and SLAM.

Dong-Wook Lee received the B.S., M.S., and Ph.D. degrees in Control and Instrumentation Engineering from Chung-Ang University, Korea, in 1996, 1998, 2000, respectively. He was a research professor at Chung-Ang University, Korea, from 2002 to 2004. He was a post-doctoral researcher at the University of Tennessee, USA, from 2004 to 2005. He is currently a principal researcher at Korea Institute of Industrial Technology(KITECH), Korea, since 2005. His research interests include android robot, social Human-Robot Interaction, and image acquisition system.