FPGA-Based Image Processor for Sensor Nodes in a Sensor Network

Masaki Yoshimura, Hideki Kawai, Taketoshi Iyota and Yongwoon Choi*

Faculty of Engineering, Soka University, 1-236 Tangimachi Hachioji, Tokyo, Japan

Abstract: A field-programmable-gate-array- (FPGA-) based image processor which can be used for sensor nodes in a sensor network has been proposed and developed. Image processors for the nodes must satisfy requirements such as low power consumption, small circuitry scale, and modifiability of the hardware architecture. By developing an image processor designed using an FPGA, SRAM modules, and the vector code correlation method which is suitable for the construction of the target hardware architecture, it was possible to ensure that the processor satisfies these requirements. In this paper, we present the details of this image processor, which employs the template matching method for target tracking as well as the background subtraction method for object extraction. In addition, in order to verify its applicability in sensor nodes, we demonstrate the usefulness of the image processor from the results of an experiment in which the template matching and background subtraction methods were implemented simultaneously.

Keywords: FPGA-based image processor, Sensor network, Template matching, Background subtraction, Real-time processing, Low power consumption.

1. INTRODUCTION

Much of the research focusing on sensor nodes that are composed of microprocessors, sensors, and actuators has been conducted in the field of sensor networks [1, 2]. Sensor nodes are linked to each other with either wired or wireless interfaces to form sensor networks, which are mounted on doors, floors, walls and other structural elements of various facilities. These sensor nodes are used for obtaining information on the environmental conditions and the behavior of the users, such as robots and humans, and the information provided by their own or other sensor nodes is used for operating and controlling their actuators. Therefore, sensor nodes in sensor networks play an essential role in implementing functions such as smart sensing and environmental monitoring [3].

The image sensors which are often used in sensor nodes due to their features, such as no contact and wide sensing range, do not necessarily satisfy the requirements for sensor nodes due to the limitations of the performance of general microprocessors in terms of power consumption and real-time response [4-6]. Accordingly, image processors which solve these problems have also been investigated from the viewpoint of hardware architecture in order to achieve real-time response [7-12]. However, all such studies have been conducted with the aim of developing effective solutions based on the limitation for the size and the type of the target images to be processed [7-9]. Furthermore, such studies have given preference to the performance-related specifications of the processors, which entail large-scale circuitry and high clock frequencies and high power consumption [9-12].

As a means to overcome these limitations, we have proposed and developed the hardware architecture of a processor which is suitable for use as an image sensor in sensor nodes. In order for an image processor to be used in sensor nodes, it should satisfy the following requirements.

1) In order to achieve low power consumption, the processor should be designed as small as possible in terms of circuitry scale, and should run as low as possible in terms of clock frequency.

2) It should be able to process images of video graphic array (VGA) size at speeds which are higher than the processing speed of general-purpose software.

3) It should be easily modifiable when designing hardware architectures for specific applications.

In order to satisfy these requirements, the image processor was built by using a static RAM (SRAM), a field-programmable-gate-array (FPGA) whose internal structure can be altered in accordance with the target application, and the vector code correlation (VCC) method, which is suitable for the construction of the desired hardware architecture [13]. In this paper, we present the details of the proposed image processor, which is used with the template matching method for target tracking and the background subtraction method for object extraction. In addition, in order to verify its applicability in sensor nodes, we also demonstrate the high performance of the image processor on the basis of the results of an experiment in which the template matching method and the background subtraction method were applied simultaneously by slightly modifying the original architecture.

2. VECTOR CODE CORRELATION

One of the widely adopted techniques for finding a target in a given image is the template matching method. This method is used to detect the target by comparing the template image of the target with all partial images of an input image. However, this common method has the disadvantage of relying on complex algorithms for finding the target in

*Address correspondence to this author at the Faculty of Engineering, Soka University, 1-236 Tangimachi Hachioji, Tokyo, Japan; Tel: +81 42 6918197; Fax: +81 42 6918197; E-mail: choi@soka.ac.jp

1876-8253/09

captured images. In order to overcome this, we have developed a kind of template matching method—the vector code correlation (VCC) method—which can efficiently improve the performance of the algorithm and the robustness against illumination changes. The VCC method is especially useful in conditions involving both illumination changes and the need for high-speed processing. VCC is a method for calculating the degree of similarity between images by comparing corresponding bits which are encoded into two bits referred to as vector code, rather than comparing the intensity of the pixels, which is a commonly used method in image processing. The vector code described here is in fact a combination of three kinds of codes for calculating the gradient of an approximate plane. The gradient of an approximate plane is composed of a square 3×3 pixels in size, which is constructed by taking into account the intensity of the 8 neighboring pixels of the target pixel, as shown in Fig. (**1**). Each pixel in the image has a vector code of 4 bits composed of a pair of the 2-bit sequences 01, 00, and 10, for both the x and the y direction. For example, the vector code for the gradient of the 3×3-pixel square in the 'template image' of Fig. (**1**) is 0101, where each vector code for the gradient is 01 (corresponding to a "positive" gradient) for both the x- and the y-direction. Likewise, the vector code for the gradients of the region given in the 'partial image' is encoded as 1001 since the sequence for the x-direction is 10 (negative), and that for the y-direction is 01 (positive), respectively. The similarity is calculated by performing an XOR operation on the two encoded vector codes, and the result in this case is 1100. By counting the number of '1's in the result (1100), we obtain the correlation value for the two vector codes, which is 2 in this case. This indicates the degree of similarity between the vector codes for two given regions. Therefore, the smaller the correlation value, the greater the similarity between the two images.

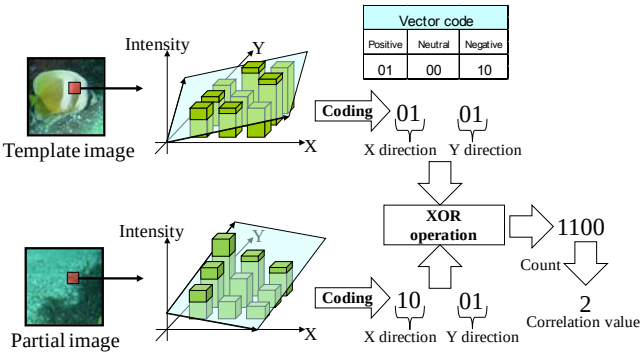

Fig. (1). Schematics of the vector code correlation method.

The VCC method for comparing two images relies on the coded gradient of each pixel instead of the intensity, and therefore it has the advantage of robustness against illumination changes. Moreover, adopting the VCC method not only reduces the amount of image data, but also simplifies the algorithm for template matching. Thus, the hardware architecture for an image processor based on VCC can be effectively designed on circuitry scale.

3. HARDWARE ARCHITECTURE

Fig. (**2**) shows the hardware architecture for an image processor based on the VCC method. This hardware architecture consists of an FPGA, including several modules and an SRAM module for storing image data. Images captured through a camera or sent *via* the communication interface by a microprocessor are stored to the SRAM. After that, the input image from the camera and the stored image are simultaneously converted into vector codes by each encoder module. The image processing block is where the internal design can be modified in accordance to the desired application. For example, if it is necessary to realize the template matching method for target tracking or the background subtraction method for object extraction with the VCC method, the image processing block is replaced with the block shown in Fig. (**3a**) or (**3b**).

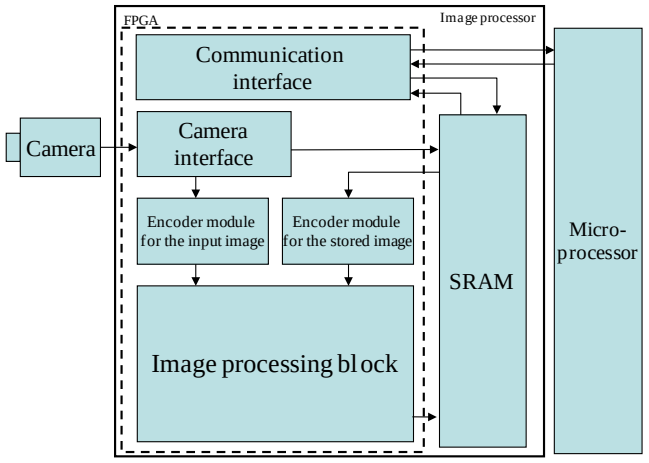

Fig. (2). Schematics of the hardware architecture based on the VCC method.

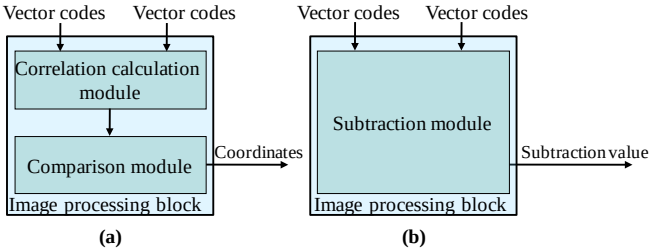

Fig. (3) Processing modules in the modifiable image processing block of Fig. (**2**): (**a**) Modules for template matching, (**b**) Module for background subtraction.

Since each module in the architecture operates and performs matching for the partial images at the frequency of the pixel clock, the processing speed for the template matching and the background subtraction methods is the same as the frame rate of the camera. Therefore, we have pursued a hardware architecture which can perform processing in real time by using sensor nodes. In this chapter, we describe the details of the modules in our designed architecture.

3.1. Encoder Module for the Vector Code

Fig. (**4**) shows the schematics of the two encoder modules, which generate the vector codes for the image inputted from the camera and the image stored in the SRAM. Although these modules are identical in terms of their function, they process image data inputted from different sources. Each module consists of four memory buffers, two selectors, 3×3 shift registers, respective filters for the x and y direc-

tions, and four comparators. Selector 1 selects one of the memory buffers and sends the intensity data of one row of the input image in a sequential manner. At the same time, selector 2 selects three memory buffers, in which the intensity data are already stored, and sends the intensity data of one pixel from each memory buffer to the shift registers by following the order of the columns. Thus, the intensity data for 3×3 pixels are stored in the shift registers. Subsequently, the stored intensity data are used for calculating the gradients through the respective filters for the x and y directions. Each gradient is encoded into a 2-bit vector code by comparing it to the thresholds of Th1 and Th2 inside the comparators. Each vector code comprises a pair of 2-bit sequences, namely 01, 10, or 00, which indicate positive, negative, and neutral gradients, respectively. Therefore, the vector code for a certain pixel is thus created with 4 bits for the x and y directions. After all vector codes for the three memory buffers are generated, selector 2 designates three memory buffers for the next row. By building the encoder module as shown in Fig. (**4**), the column intensity data for 3×3 pixels is stored at a rate corresponding to the pixel clock frequency, and therefore the input image can be converted into vector codes at a rate which is the same as the frame rate of the camera.

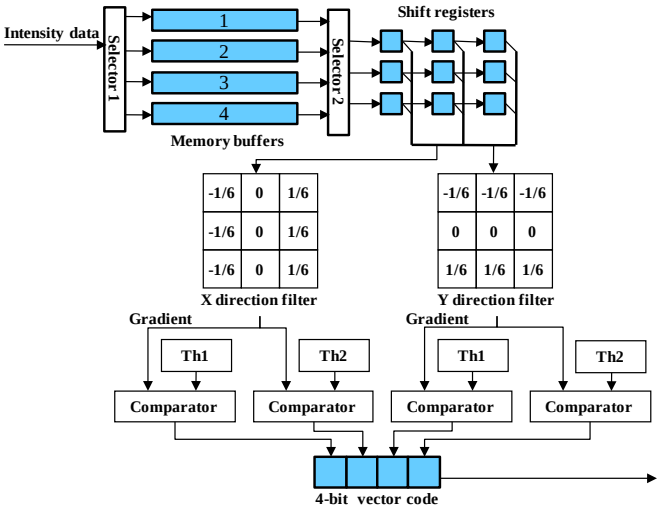

Fig. (4). Details of two encoder modules shown in Fig. (**2**).

3.2. Architecture for Template Matching

The template matching method is implemented by using the VCC method on the FPGA with the hardware architecture connected to the modules in the image processing block, as shown in Fig. (**3a**). The correlation calculation module generates the correlation values, which are obtained by counting the number of '1's in the result of the XOR operation applied to the vector codes of the partial image and the template image, which are converted by the two encoder modules. In the comparison module, the correlation value of a given location is compared with that of another location in order to obtain the minimum correlation value. After performing a comparison with all correlation values for the input image, the comparison module sends the minimum correlation value and the coordinates of the target location to the SRAM, after which those data are transmitted to the microprocessor through the communication interface. The details of each module for the template matching method are explained in the following sections.

3.2.1. Correlation Calculation Module

Fig. (**5**) shows the schematics of the correlation calculation module, which is used for template matching between a given template image and one of the partial images of an input image. The size of both the template and the partial image is 32×32 pixels. This module is composed of two clusters of 32×32 shift registers (SR1 and SR2), a 31-unit FIFO buffer, and XOR and SUM operator modules. The vector codes encoded by the two encoder modules are sequentially inputted to the SR1 and SR2 shift registers, starting with the upper left corner of the respective images.

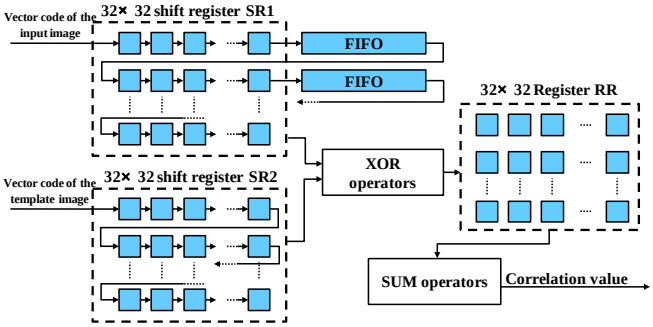

Fig. (5). Details of the correlation calculation module shown in Fig. (**3a**).

In particular, the vector codes of 32 rows of the input images are first stored to the respective registers of 32 rows in order to form a partial image in the 32×32 SR1 shift registers, and the remainders in each row for the input image are saved to the respective 31-unit FIFO buffer. Here, the FIFO buffer only has the capacity to subtract the vector codes of the 32 shift registers from the vector codes of a single row of the input image. Therefore, when the vector codes completely occupy the shift registers of SR1, each XOR operator generates the result for the vector code corresponding to the same position of the registers in SR1 and SR2, sending it to the RR register. The number of '1's (the correlation value) in the results saved in the RR register is calculated by the SUM operators and is outputted to the comparison module. Thus, since the flow in this module is implemented in parallel at the pixel clock frequency over the entire constituent elements forming the module, the processing speed with respect to the input image matches the camera frame rate.

3.2.2. Comparison Module

Fig. (**6**) shows the schematic details of the comparison module. Register 1 in the comparison module stores the first correlation value from the correlation calculation module. The value of register 1 is compared to the next correlation value in the comparator, which creates a signal named "comp signal". Depending on the resulting comp signal, selector 1 sends the smaller correlation value to register 1. Subsequently, the values of the counters for the x and y directions are stored to register 2 and register 3, respectively, which represent the x and y coordinates of the pixel with the smaller correlation value. After comparing all input correlation values of an image, the correlation value and the x and y coordinates are stored to register 4 and are outputted to the microprocessor. The partial image with the minimum correlation value processed on the basis of the input image indi-

cates the target, and corresponding the x and y coordinates indicate its position.

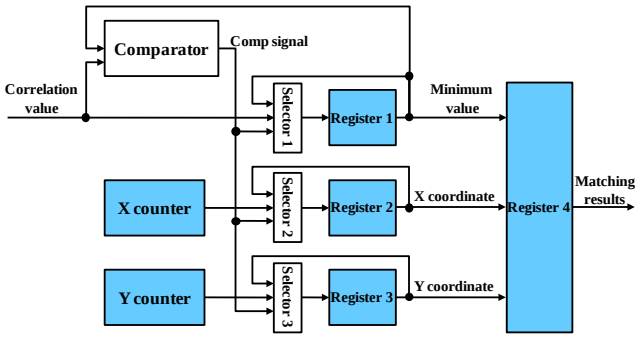

Fig. (6). Details of the comparison module illustrated in Fig. (**3a**).

3.3. Architecture for Background Subtraction

The background subtraction method is used for extracting only the changed portions of the image by subtracting the previously obtained background image from the input image. Fig. (**7**) shows a notional diagram which explains the background subtraction method, which is based on the VCC method. The background subtraction method is implemented by comparing all partial images of the same location of the input image and the given background image. For example, if the results for two partial images with coordinates (x1, y1) in both images are different, it is considered that an 'object' has appeared in the background image. If the results are similar, as shown in location (x2, y2), the partial image is classified as 'background'. This method can be implemented by replacing the image processing block of Fig. (**2**) with the subtraction module shown in Fig. (**8**).

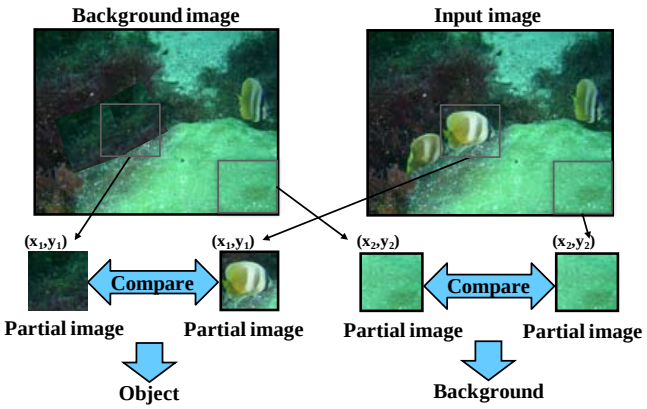

Fig. (7). Notional diagram explaining the background subtraction method.

Fig. (**8**) shows the schematics of the subtraction module, which is composed of two clusters of 8×8 shift registers (SR1 and SR2), two FIFO buffers comprising 7 units each, XOR and SUM operators, and a comparator. This module is prepared on the basis of a similar idea to that of the module in Fig. (**5**), which is described in Section 3.2.1. However, the shift registers for the partial images are formed with respect to 8×8 pixels, as opposed to the 32×32 registers in SR1 and SR2 of Fig. (**5**). Accordingly, each FIFO in this module has the capacity to store the vector codes of 7 pixels for one row of the input image.

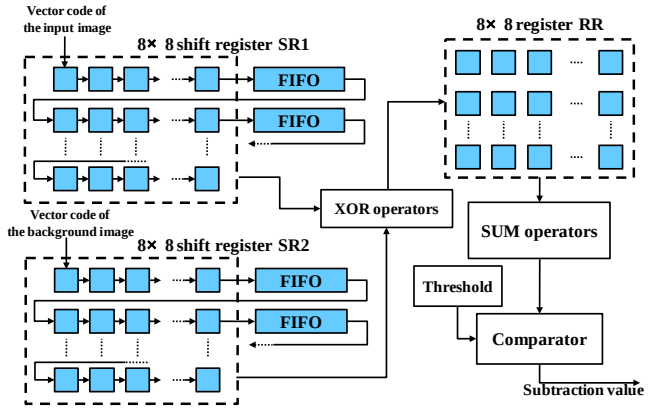

Fig. (8). Details of the subtraction module illustrated in Fig. (**3b**).

The vector codes from the two encoder modules are simultaneously inputted to the SR1 and SR2 shift registers, respectively. When the vector code for each upper left pixel of the partial images is stored to the respective bottom right shift registers of SR1 and SR2, 8×8-pixel partial images are formed in SR1 and SR2 for the same location in both the input image and the background image, and the result of the XOR operation is in turn stored to the RR registers at each clock cycle. The correlation values for the stored results are calculated by using the SUM operators and are compared to a threshold in the comparator at each clock cycle. The result of the comparison is outputted as the subtraction value, where a subtraction value of '1' indicates the presence of an 'object' if the correlation value is equal to or greater than the threshold, and '0' indicates 'background' if the correlation value is smaller than the threshold.

4. EXPERIMENTS

4.1. Experimental Environment

In order to verify the running state of the hardware architectures designed as shown in Figs. (**2**) and (**3**), we performed target tracking and object extraction experiments in an indoor environment. The camera shown in Fig. (**9**) has the

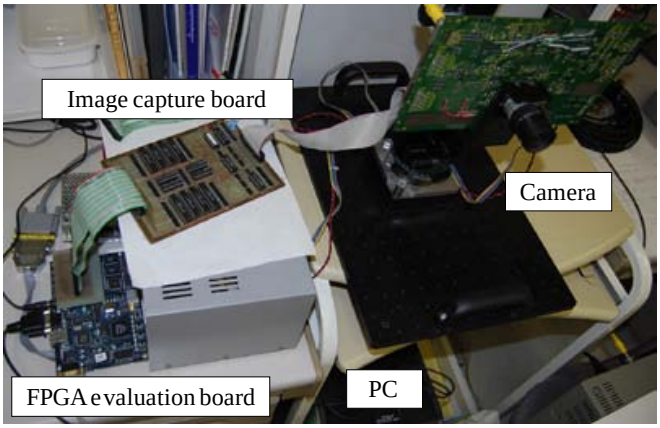

Fig. (9). Setup used to verify the running state of our hardware architecture.

following parameters: CCD 60fps, 640×480 pixels/frame, and 8bits/pixel. The operation clock of the FPGA used here was equal to that of the camera, which is 25MHz. The image capture board connected to the camera is linked to the FPGA evaluation board. For the purpose of evaluating the designed hardware architecture, we used a Cyclone FPGA and a serial port for communication between the FPGA and the PC.

4.2. Target Tracking Experiment

We performed an experiment for target tracking based on the template matching method using the VCC method. Fig. (**10**) shows an input image and the template image of a target in the image. The template image for target tracking, which is 32×32 pixels in size, is in the middle of the input image. By horizontally moving the target to the right or to left in stages, it was confirmed that the matching accuracy was high and the processing speed matched the frame rate of a camera. Fig. (**11a**) and (**11b**) illustrate the trace of the x and the y coordinates for the central position of the target in the image, respectively. For example, the graph in Fig. (**11a**) indicates that the target is in a steady state with respect to the horizontal axis of the stages corresponding to frames 0-170 and 700-900, and that it is moving to the left in frames 170-320 and 550-700 and to the right in frames 320-550. Fig. (**11b**) shows that there is no significant movement along the y axis in the image since the trace is almost leveled. From the experimental results, we concluded that the matching accuracy is within ±5 pixels of the acceptable range and that the processing time is 1/60 seconds per frame.

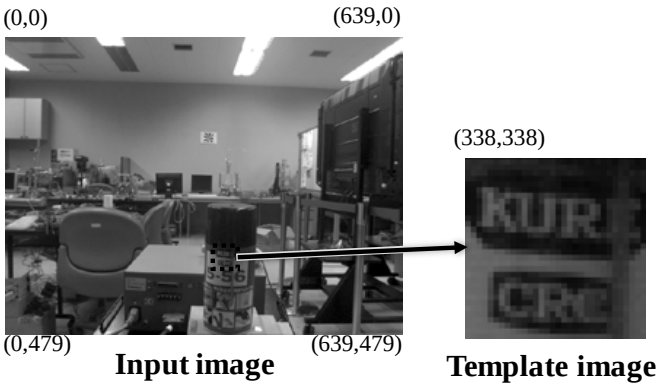

Fig. (10). Input and template images used in the experiment.

4.3. Object Extraction Experiment

We performed experiments for object extraction by using the background subtraction method, which was implemented on our hardware architecture. In these experiments, we attempted to evaluate whether a human body as an object can be detected and extracted as a roughly outlined region. Fig. (**12**) shows one of the experimental results for object extraction. The image shown in Fig. (**12c**) was obtained by subtracting the background image from the input image, which are shown in Fig. (**12a**) and (**12b**), respectively. As demonstrated by the experimental result in Fig. (**12c**), although some parts of the object were not detected, it was confirmed that the rough shape of a human body can be extracted. This means that an area corresponding to a human body can be obtained when using the image sensor on the sensor nodes.

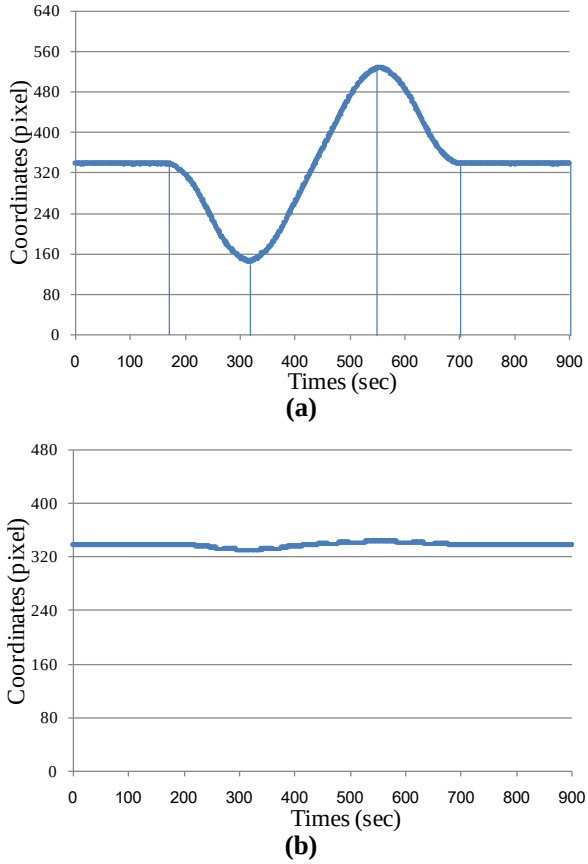

Fig. (11). Experimental results for template matching: (**a**) Transition of the x coordinates in accordance with the stage, (**b**) Transition of the y coordinates in accordance with the stage.

Additionally, there was mp undesirable noise in the subtracted image, which implies that our hardware architecture can be used in small-scale image sensors since there is no need for additional complicated algorithms for noise elimination. From the experimental results, we confirmed that it is possible to use our hardware architecture for object extraction, as well as that its performance is sufficiently high for the implementation of the background subtraction method by using small-scale image sensors.

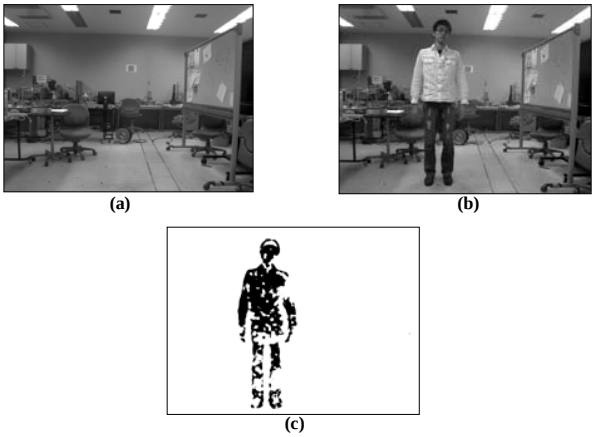

Fig. (12). Images used in the experiment and the image resulting from the background subtraction experiment: (**a**) Background image, (**b**) Input image, (**c**) Subtracted image.

Table 1. Circuitry Scale and Power Consumption of the FPGA (Cyclone III: EP3C25)

	Template Matching	Background Subtraction
Logic elements	18,533 (75%)	2,819 (11%)
Internal memories	163,840 bits (27%)	122,880 bits (20%)
Power consumption	229mW	165mW

4.4. Circuitry Scale and Power Consumption

Table **1** shows the dimensions of the circuitry and the estimated power consumption of our hardware architecture, which was implemented with the latest Cyclone III (EP3C25) FPGA. The size of this FPGA is moderate, and it runs with lower power consumption than other FPGA device families. Our architecture for the template matching and background subtraction methods was realized with FPGA ratios of 75% and 11% for the logic elements and 27% and 20% for the internal memory buffers, respectively. From these results, we confirmed that these methods can be implemented on a typical FPGA on circuitry scale. Also, the estimated power consumption for executing these two tasks was 229mW and 165mW, respectively. As a power consumption of 1W or less is usually considered low, the developed architecture can be considered to have satisfied the requirements for power consumption.

5. PERFORMANCE

An experiment was conducted in order to verify the applicability of the designed hardware in real indoor environments. The experiment consisted of tracking the region of the body and the face of a person in a captured image. This might be useful for the detection of persons entering and leaving a room, for example, in the field of monitoring and surveillance. The purpose of this experiment was to examine whether our hardware can be used in sensor nodes in the field of sensor networks, and whether it is suitable in terms of scale and performance for implementing the template matching and background subtraction methods, which are often used in this field. However, in order to perform this experiment, we added the modules shown in Fig. (**13**) to the image processing block illustrated in Fig. (**2**). We provide an explanation of the details regarding the added modules in the following sections.

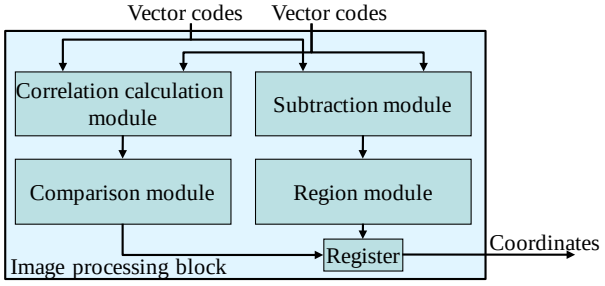

Fig. (13). Modules in the image processing block of Fig. (**2**) designed for the current experiment.

5.1. Hardware Architecture for this Experiment

The schematics of the additional modules in the image processing block necessary for implementing the template matching and background subtraction methods simultaneously are shown in Fig. (**13**). The input image and the stored image, which are encoded as vector codes, as shown in Fig. (**2**), are inputted to both the correlation calculation module and the subtraction module. The input data are processed in the same manner as explained in Sections 3.2 and 3.3, respectively. As a result, the minimum and the maximum coordinates for the entire body and face regions of a person are outputted into the register. Fig. (**14**) shows the schematics of the region module illustrated in Fig. (**13**).

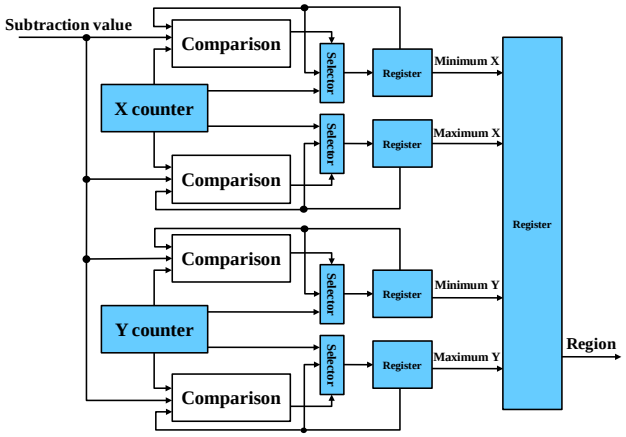

Fig. (14). Details of the region detection module.

This module consists of two counters for the x and y coordinates, four comparators, four selectors, and four registers. The registers store the minimum or the maximum coordinates for the x and y axes by comparing the coordinates of each counter for the region formed when the subtraction value is regarded as an object. After comparing it to all subtraction values for an image, the stored coordinates corresponding to the region of the target object are outputted to the PC.

5.2. Results of the Experiment

The result of the simultaneous implementation of the template matching method with the template image in Fig. (**15a**) and the background subtraction method with the background image in Fig. (**15b**) is shown in Fig. (**15c**). It is clear from Fig. (**15c**) that the large rectangular region surrounding the entire body and the small rectangle inside it correspond to the region of the object extracted with the background subtraction method and the location of the human face detected with the template matching method, respectively. From these results, we know that the two methods can be implemented at a rate matching the camera frame rate with the hardware architecture constructed on one FPGA used in our experiments. Regarding the FPGA used in this experiment, the usage rate for the logic elements, the internal memories, and the estimated power consumption was 85%,

37%, and 242mW, respectively, as shown in Table **2**. This means that the hardware architecture is useful in terms of power consumption, circuitry scale, and processing speed as required for the sensor nodes in a sensor network. In particular, the obtained processing speed suggests that our hardware architecture exceeds the processing speed of general-purpose software.

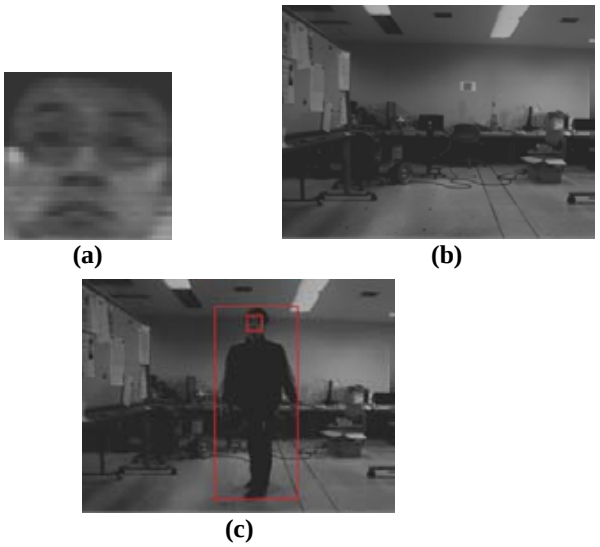

Fig. (15) Excerpt of the results of the experiment combining the two methods: (**a**) Template image, (**b**) Background image, (**c**) Resulting image.

Table 2. Circuitry Scale and Power Consumption on the FPGA (Cyclone: EP3C25) for the Application

	Application
Logic elements	20,932 (85%)
Internal memories	225,280 bits (37%)
Power consumption	242mW

6. CONCLUSION

We have proposed an image processor which can be used in the sensor nodes of a sensor network. Image processors on sensor nodes must satisfy the requirements of power consumption, circuitry scale, and modifiability of the hardware architecture. We successfully developed a hardware architecture which satisfies these requirements by developing an image processor including a FPGA and SRAM. In terms of power consumption and circuitry scale, a solution was found in constructing the hardware architecture by adopting the vector code correlation (VCC) method. The requirements for our hardware architecture were satisfied for a camera frame rate of 60fps with respect to the processing time and the low power consumption of 242mW. We performed simultaneous target tracking and object extraction with respect to an image in order to verify that the requirements were satisfied. In addition, from for the viewpoint of practical use, we confirmed that the processor based on our hardware architecture has the advantage of exceeding the speed of general-purpose software with respect to processing VGA images.

7. ACKNOWLEDGEMENT

This research was partially supported by the "Collaboration with Local Communities" Project for Private Universities on "Development of Ubiquitous Monitoring Network Based on Distributed Sensor Nodes using Local Positioning / Optical Sensory Nerves and their Industrial Applications", a matching fund subsidy from MEXT (The Ministry of Education, Culture, Sports, Science and Technology of Japan), 2006-2010.

REFERENCES

[1] F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey", *Comput. Newt.*, vol. 38, pp. 393-422, March 2002.

[2] M. A. M. Vieira, C. N. Coelho Jr, D. C. da Ssilva Jr. and J. M. da Mata, "Survey on wireless sensor network devices", in 9th IEEE International Conference on Emerging Technologies and Factory Automation, 2003, pp. 537- 544.

[3] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks", *Computer*, vol. 37, pp. 41-49, August 2004.

[4] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: in situ image sensing and interpretation in wireless sensor networks", in 3rd International Conference on Embedded Networked Sensor Systems, 2005, pp. 192-204.

[5] S. Hengstler, and H. Aghajan, "WiSNAP: a wireless image sensor network application platform", in 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006, pp. 6-12.

[6] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan, "MeshEye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance", in 6th International Conference on Information Processing in Sensor Networks, 2007, pp. 360-369.

[7] T. T. Kwok, and Y. K. Kwok, "Computation and energy efficient image processing in wireless sensor networks based on reconfigurable computing", in International Conference on Parallel Processing, 2006, pp. 43-50.

[8] H. Matsubayashi, S. Nino, T. Aramaki, Y. Shibata, and K. Oguri, "Retrieving 3-D information with streamed template matching", in Technical Committee on Reconfigurable Systems, 2007, pp. 19-24.

[9] I. Ohmura, O. Mitamura, K. Nakahara, H. Takauji, S. Kaneko, M. Shimizu, and Y. Miyashita, "A real time velocity sensor for agri-motors by use of FPGA realization of orientation code matching", *IEICE Trans. Info. Syst.*, vol. J91-D, pp. 1325-1335, May 2008.

[10] J. Gause, P. Y. K. Cheung, and W. Luk, "Reconfigurable shape-adaptive template matching architectures", in IEEE Symposium on Field-Programmable Custom Computing Machines, 2002, pp. 98-107.

[11] S. Hezel, A. Kugel, R. Manner, and D. M. Gavrila, "FPGA-based template matching using distance transforms", in IEEE Symposium on Field-Programmable Custom Computing Machines, 2002, pp. 89-97.

[12] C. T. Huitzil, and M.A. Estrada, "FPGA-based configurable systolic architecture for window-based image processing", *EURASIP J. Appl. Signal Process.*, vol. 2005, pp. 1024-1034, July 2005.

[13] M. Yoshimura, H. Kawai, T. Iyota, and Y. Choi, "Hardware design of vector code correlation method for high-speed template matching", in International Conference on Control Automation and Systems, 2008, pp. 2529-2532.

Received: December 24, 2008 Revised: February 19, 2009 Accepted: February 25, 2009