Journal of Computer Sciences and Applications, 2015, Vol. 3, No. 1, 1-10
Available online at http://pubs.sciepub.com/jcsa/3/1/1

DOI:10.12691/jcsa-3-1-1

Self-Learning of Feature Regions for Image Recognition

Satoru Yokota[1,*], Jiang Li[1], Yuichi Ogishima[1], Hiromasa Kubo[1], Hakaru Tamukoh[2,*], Masatoshi Sekine[1]

[1]Graduate School of Engineering, Tokyo University of Agriculture and Technology, Tokyo, Japan
[2]Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology, Kitakyushu, Japan
*Corresponding author: yokota@sekine-lab.ei.tuat.ac.jp, tamukoh@brain.kyutech.ac.jp

Received January 09, 2015; Revised January 19, 2015; Accepted January 22, 2015

Abstract Mobile systems are used in various environments. Thus, it is practical for image recognition systems to autonomously learn template images that are adaptive to objects in their various environments. However, learning the features of such objects requires large-scale computation and complex control. Hence, we propose an image recognition system that selects and learns regions that have a given object's features. This system is designed as a hardware/software (hw/sw) complex system with the multi-dimensional field programmable gate array (FPGA) "Vocalise." This study discusses the possibility of dynamically building image databases and of real-time learning using the proposed image recognition system. Results indicate that the learning speed of the proposed method is estimated to be 1.4×10^3 faster than that obtained with a conventional software method. This suggests the possibility of real-time learning.

Keywords: *autonomous learning, feature region, hw/sw complex system, image recognition, vocalise*

Cite This Article: Satoru Yokota, Jiang Li, Yuichi Ogishima, Hiromasa Kubo, Hakaru Tamukoh, and Masatoshi Sekine, "Self-Learning of Feature Regions for Image Recognition." *Journal of Computer Sciences and Applications*, vol. 3, no. 1 (2015): 1-10. doi: 10.12691/jcsa-3-1-1.

1. Introduction

Of the many different intellectual processes available, image recognition systems have drawn public attention in recent years as a technology necessary for vehicular accident prevention, autonomous cars, and AI robots. Given the fact that use of mobile systems (such as cars and robots) with image recognition systems will increasingly spread in various environments, an image recognition system with a learning function that can easily adapt to various situations in real time is indispensable.

In order to achieve this type of intelligent processing, close cooperation between software and logic circuits in a system large-scale integration (LSI) is essential to quickly process vast amounts of data. The Viola–Jones method [1], which focuses on the brightness of sub-regions (Harr-like features) as facial features, is known for having an accurate and rapid face-detection algorithm and learns features by using AdaBoost [2]; This method is able to detect faces from 384 × 288 pixel images on a 700 MHz Pentium III processor in about 67 ms (15 fps). Using this method, 38-layer detectors were trained on a single 466 MHz AlphaStation XP900 with a huge number of images that spanned weeks; even though the algorithm was paralleled, it took about a day to process through all the images. In other words, real-time learning using this method is difficult due to the many layers and training images involved. Although the detection seems to be processed in the software in nearly real-time using the Viola–Jones method, the execution time increases with the increasing number of pixels. Even optimized OpenCV took 561 ms (1.78 fps) to execute detection processing of 627 × 441 pixel images using the Viola–Jones method [3].

Studies of pedestrian detection have also been popular in recent years. Human bodies vary more in color, brightness, and shape when compared with human faces. Thus, for such studies to be successful, a feature more robust to the variations is required. Combining the histograms of oriented gradients (HOG) feature with a support vector machine (SVM) has been proposed [4] and has been proven to be more robust than the Haar-like feature in the Viola–Jones method, but its detection speed is just less than 1 [fps] on high-end CPUs [5].

Thus, image recognition using a conventional, software-only approach requires a lot of time, and realizing real-time processing with such an approach is extremely difficult. In addition, when an intellectual process is conducted as part of the system, the system is required to utilize the result of that process and execute successive processes, which easily increases the time required for execution. Therefore, parallel and pipeline processing become indispensable to intellectual processing.

With uniform (homogeneous) computing, method processing could be considered in parallel on a uniform processor array, i.e., general purpose computation on graphics processing units. However, when different processes are mixed (heterogeneous computing), to execute parallel processing must be executed using the logic circuits that correspond to these processes. On the other hand, professional knowledge is necessary for parallel processing of software programs and logic circuits. To settle this issue, the system must be designed such that the logic circuits are encapsulated within the application software, thus allowing them to be easily operated. To

realize this objective, we have previously proposed a logic system that was designed by objectifying logic circuits, called the hardware/software (hw/sw) complex system [6].

By applying the concept of the hw/sw complex system, we have also proposed various intellectual processing circuits (such as for image recognition [7] and voice recognition [8]) with the aim of integrating these circuits into a mobile system. Because mobile systems like robots require low power consumption and compact size, only weak processors can be loaded onto them. However, even a system with a weak processor can be easily accelerated by logic circuits. To examine this possibility, we used a logic circuit to complete phoneme processing of a voice recognition application and integrated the logic circuit with the recognition component (post-process) of conventional voice recognition software [9]. Also, we simultaneously integrated a vehicle control circuit [10].

Furthermore, we have utilized template matching as an image recognition method because of its convenience [11], and we have utilized a self-organizing map (SOM), which is a kind of machine learning, as a method for making templates [12]. We have also investigated a method to extract robust features by hierarchically learning multi-resolution images using a tree-structured SOM; however, as we move to the finer-image tree layer, the amount of data to learn increases exponentially, which means that the amount of learning time required is very long.

Therefore, we turned our focus to the variance (i.e., statistical quantity) of images and proposed a method for reducing the required computational amount [13,14]. An experiment showed that this method can selectively learn characteristic regions (such as chins or noses) and reduced data amounts to within 1/2 to 1/10th that of the previous method in each layer [15]. We then calculated SOM learning with a lot of computational complexity by using one field programmable gate array (FPGA) board in order to increase learning times; however, the combined system of SOM learning executed on one FPGA board, and feature-region search executed on a host CPU did not perform well due to a bottleneck in the communication between the FPGA board and host CPU. Thus, we found that this system could not learn in real time and increasing the parallel degree of tree-structured SOM learning was difficult because one FPGA board did not have enough capacity to realize plural SOM circuits.

Hence, we designed new circuits not only for SOM learning but also for the feature-region search that had run on the host CPU in the previous system. In addition, we designed an image recognition system for self-learning of feature regions that is implemented on a multi-dimensional FPGA, the "Virtual Object by Configurable Array of Little Scalable Engine" (Vocalise). The new circuits reduce communication between the host CPU and FPGA board. They also enable the parallel processing of the sub-regions of any given image required for the feature-region search. Implementation of the system on Vocalise allows us to load many SOM circuits and increases the degree of parallelism. We think that these modifications will enable real-time learning.

Furthermore, dynamic building of image databases is possible, which means that knowledge of objects can be automatically acquired on demand, thus enabling a mobile system to learn about obstacles by passing scenes while running. Here, the term "image database" refers to any tree-structured feature region template. Any object can be represented using an interrelated set of feature regions.

This study proposes an image recognition system for learning feature regions and evaluates its performance. With the proposed method, learning speeds are expected to increase by 1.4×10^3 times when compared with software-only systems, and recognition speeds are expected to increase by 13 times faster when compared with software implementation. This study is organized as follows. Section 2 explains the proposed method for feature-region learning. Section 3 explains the method's configuration (or realization). Section 4 discusses the method's performance evaluation as well as the circuit scale, power consumption, and possible speeds of a mobile system. The conclusions are given in Section 5.

2. Feature-Region Learning Method

Feature-region learning is a method for building tree-structured feature region templates by learning features that have been narrowed down using multiple layers and resolutions — in other words, by generating multi-resolution images by wavelet transform (here, the Haar transform) of the input images. Let the coarsest images be the root and finer images be branches in order to make a multi-resolution image tree with a hierarchical structure. In images of each resolution, there are plural sub-regions with characteristic images. Using the images in these sub-regions creates templates that express an image characteristic. These templates can expand (express) an image. Likewise, these templates have coarse to fine tree-structure relationships, which can be built by combining the feature-region search with the tree-structured SOM.

2.1. Feature-Region Acquisition Method

2.1.1. Feature Regions and the Acquisition Algorithm

Feature regions can be classified into the two classes: high-variance regions and low-variance regions. High-variance regions are those whose differences among images are easily seen; low-variance regions are those whose differences among images hardly appear.

The proposed method for searching feature-regions is as follows:

1: Select the feature region of an input image and acquire the base vectors from the selected area.

2: Expand the input images in the selected area using the base vectors.

3: Acquire the distribution variance of the expansion coefficients.

4: Repeat steps 1–3 for each input image in that layer. Let regions where the variance is high be grouped as high-variance regions. Let regions where the variance is low be grouped as low-variance regions.

5: Selectively learn the feature regions of images in the next layer.

In the proposed system, the cluster centers acquired by SOM learning are used as base vectors for simplicity.

2.1.2. The SOM Learning Method

This section describes a SOM learning method that includes clustering. SOM is an unsupervised type of classification learning for representing the input data of a

multi-dimensional space into a low dimensional map. It updates reference vectors placed in the low dimensional map according to input data. Clustering reference vectors in the trained map provides cluster centroids that can expand (express) input data.

The learning procedure is shown below:

1: Initialize reference vectors $\vec{w}_j$ by random data.

2: Input data (input vectors $\vec{x}$) that are randomly picked from the target data to the SOM.

3: Calculate the distance between an input vector $\vec{x}$ and a reference vector $\vec{w}_j$.

4: Let the reference vector with the smallest calculated distance be the winning vector *c*.

5: Update such that the winning vector and its neighborhood vectors are close to an input vector, as shown in Equation 1:

$$\vec{w}_i(t+1) = \vec{w}_i(t) + h_c(t)\alpha(t)(\vec{x} - \vec{w}_i(t)) \quad (1)$$

where t is the time and, α is the learning rate. The neighborhood vectors are determined by the neighborhood function $h_c(t)$.

6: Repeat steps 2–5 until the updated reference vectors converge.

7: After calculating the neighborhood distance, let the consecutive reference vectors whose neighborhood distances become less than the threshold be one cluster (clustering).

8: Let the mean vector of all reference vectors belonging to the same cluster be the cluster centroid.

During circuit implementation, the operation of the learning rate is realized by bit shift, and the neighborhood function is realized by a look-up table to reduce the circuit scale. In addition, let the clusters at the four corners of the map be the cluster centroids without clustering because cluster centroids often empirically appear in the corners.

2.2. Acquisition of the Tree-Structured Template

2.2.1. The Tree-Structured SOM

Let the tree-structured SOM be *SOMTree*, defined as Equation 2. An overview of the tree-structured SOM is shown in Figure 1.

$$\begin{aligned} &SOMTree = \{SOM^1; \\ &SOM_1^2, \cdots, SOM_i^2, \cdots SOM_I^2; \\ &SOM_{11}^3, \cdots, SOM_{ij}^3, \cdots, SOM_{IJ_I}^3; \\ &SOM_{111}^4, \cdots, SOM_{ijk}^4, \cdots, SOM_{IJ_IK_{IJ_I}}^4 \} \end{aligned} \quad (2)$$

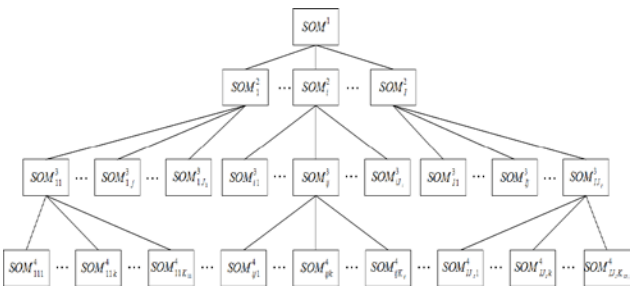

Figure 1. Overview of tree-structured SOM

Tree-structured SOM building is shown in Figure 2; its procedure is as follows:

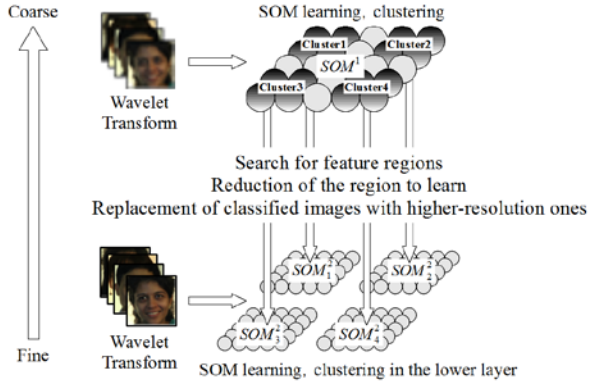

Figure 2. Making child nodes and inputting images into the tree-structured SOM

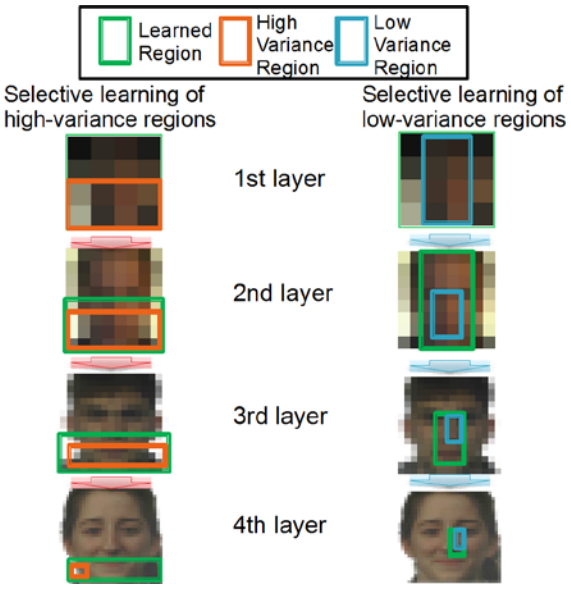

Figure 3. Feature regions acquired by feature-region learning

1: Apply wavelet transform to input images for multi-resolution analysis.

2: Input low-resolution images into the first layer SOM^1.

3: Cut out the feature region of the input images and learn that region.

4: Apply clustering to the reference vectors in the trained map and classify input images by their distance from each cluster centroid $CC_{li}^n (i = 1, 2, \cdots, N_{cl}^n)$ $(i = 1, 2, \cdots, N_{cl}^n)$.

5: Apply Gram–Schmidt orthonormalization to the cluster centroids to make base vectors $\vec{u}_{li}^n (i = 1, 2, \cdots, N_{cl}^n)$.

6: Search feature regions using base vectors to acquire them (see Section 2.1.1.).

7: Build the same number of SOMs in the next layer as there are cluster centroids N_c^1 made by SOM^1 (i.e., the number of cluster centroids provided by a SOM in the present layer).

8: Replace the input images classified in the present layer with finer resolution images and let these replaced images be the input for the SOMs in the next layer, corresponding to each cluster.

9: Repeat the procedure for the rest of the SOMs in *SOMTree*.

In the proposed system, Gram–Schmidt orthonormalization is omitted, and cluster centroids are constantly at the four corner.

The region reduction shown in Figure 1 is realized through selective learning of the feature regions, which can shorten the enormous processing time required. In addition, the result of feature-region learning for facial images [16,17] is shown in Figure 3. We can see that this method automatically acquires features considered by humans, such as chins, and can express a facial image by using the mutual relations of the feature regions.

2.2.2. Template-Making Method

This section describes a method to make templates using base vectors. A template is a linear combination of base vectors and expansion coefficients. A template is freely made by changing the value of the expansion coefficients. Templates are tree-structured just like the tree-structured SOMs. In the proposed system, a cluster centroid is considered a template for simplicity.

2.3. The Image Recognition Method

This section describes matching operations and a matching method that uses tree-structured templates.

2.3.1. Matching Operation

Template matching is conducted using the sum of absolute differences (SAD) as the distance between images. Equation 3 shows the general equation of the sum of absolute differences with template $T_{x,y}$ and detection window $I_{x,y}$.

$$M = \sum_{h=0}^{H}\sum_{w=0}^{W}|T_{w,h} - I_{x+w,y+h}| \quad (3)$$

$$M = \sum_{h=0}^{H}\sum_{w=0}^{W} Dist_{w,h} \quad (4)$$

Equations 4–8 show the matching operation of images in the YUV (brightness and color) representation. Here, K_y, K_v, and K_u represent the degree of contribution. When matching facial images, K_y = 1.0, K_v = 4.0, and K_u = 4.0 to increase the contribution of skin color [11]. W and H show the width and height of the template, respectively.

$$dY_{w,h} = \left|Y_T(T^n_{w,h}) - Y_I(I^n_{w,h})\right| \quad (5)$$

$$dU_{w,h} = \left|U_T(T^n_{w,h}) - U_I(I^n_{w,h})\right| \quad (6)$$

$$dV_{w,h} = \left|V_T(T^n_{w,h}) - V_I(I^n_{w,h})\right| \quad (7)$$

$$Dist_{w,h} = \left|K_y dY_{w,h} + K_u dU_{w,h} + K_v dV_{w,h}\right| \quad (8)$$

2.3.2. Tree-Structured Template Matching

Tree-structured template matching is coarse-to-fine template matching wherein the resolution of the template changes according to a hierarchy. Matching between high-resolution images reflects the result of matching between low-resolution images. The advantage of coarse-to-fine template matching is its reduction of computational complexity and noise.

The matching procedure is as follows:

1: Execute template matching using templates from the first layer and beginning with input images that have low-resolution.

2: Substitute input images with higher-resolution images and let them be input images for the second layer.

3: Let the scanned area in the second layer be the area in and around the template-matched area (the sum of absolute differences is minimal).

4: Execute template matching using the second-layer templates.

5: Repeat the procedure until the bottom layer is reached.

Narrowing the scanned area in this way reduces the computational amount required for matching operations.

3. Configuration Method

3.1. The hw/sw Complex System

An hw/sw complex system is a system in which the hardware and software cooperate. It consists of FPGA board(s) and a host CPU, which enables both the circuit's high-speed parallel distributed processing and the software's flexible processing. In this system, we call a circuit that executes operations in FPGA an "hwNet," an object that encapsulates the hwNet an "hwObject," and an object executed by the host CPU a "swObject." As we can control hwNets by sending messages toward hwObjects in the application software, controlling hardware and software is much less complicated than in other types of systems.

3.2. Vocalise

Figure 4 shows an overview of Vocalise, which consists of a large number of hwModuleVSes (VSes). A VS has connectors in six directions, two synchronous dynamic random access memories (SDRAMs), and one FPGA (Xilinx Spar-tan3 XC3S4000). Vocalise's scalable design allows us to change the number of VSes to fit the operational scale. An FPGA board used to control Vocalise is connected to the host CPU through a Peripheral Component Interconnect (PCI) slot, which can easily realize large-scale calculations requiring a large number of circuits.

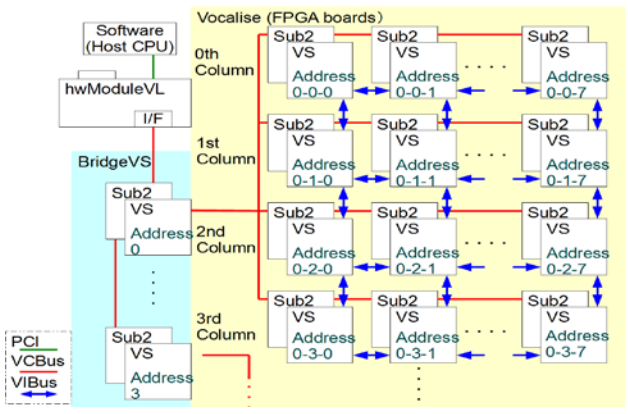

Figure 4. Overview of Vocalise

3.3. Feature Learning / Image Recognition System

Figure 5 shows the an overview of a recognition system as it learns feature regions by using Vocalise. Learning and recognition processes utilize the same hwNet because recognition processes can reuse a feature-region learning (FRL) circuit's distance calculation circuit (Section 3. 4. 1), which is loaded as an hwNet. However, the input data must be changed when reusing a learning circuit as a recognition circuit. We accomplish this by using two different kinds of hwNet encapsulation: feature learning hwObject and image recognition hwObject.

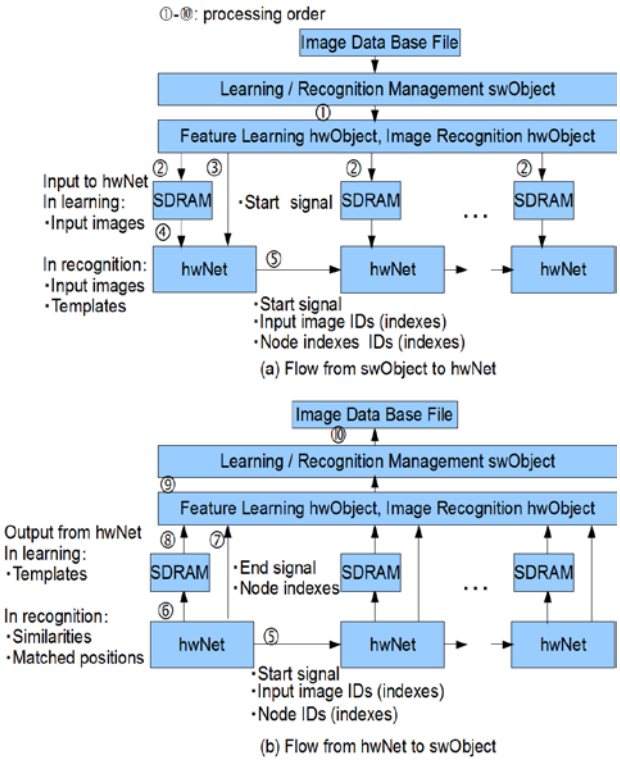

Figure 5. Overview of a feature learning / image recognition system and its processing flow

The learning flow is as follows:

1: A user pushes the learning start button.

2: The learning / recognition management swObject commands the feature learning hwObject to start learning.

3: The hwObject transmits input images to SDRAMs (when the system is loaded into a mobile system, we implement an hwNet that transmits the input from a camera attached to the mobile system).

4: The hwObject commands the first learning / recognition hwNet to start.

5: The hwNet starts to learn (i.e., acquires templates and feature regions).

6: The hwNet sends input image IDs, child node IDs, and a start signal to an hwNet of the next layer.

7: The hwNet writes templates to SDRAMs.

8: The hwNet transmits node IDs and a learning end signal to the feature learning hwObject.

9: The hwObject reads the templates from the target SDRAM.

10: The swObject gives each template a file name (i.e., a node ID).

11: Repeat steps 5–10 in the hwNet of the next layer.

Only the first node receives a start signal from a hwObject (step 4), and hwNets give direct commands (e.g., start signals) to the remaining nodes, thus avoiding communication bottleneck between hardware and software.

The recognition flow is as follows:

1: A user pushes the recognition start button.

2: The learning/recognition management swObject commands the image recognition hwObject to start recognizing.

3: The hwObject transmits templates to an SDRAM.

4: The hwObject commands the first learning/recognition hwNet to start.

5: The hwNet starts to recognize (i.e., acquires the template and position when maximum similarity to an input image occurs.

6: The hwNet transmits an input image ID, node IDs, and a start signal to a learning/recognition hwNet of the next layer.

7: The hwNet transmits its node ID, the template and position (that are acquired in step 5) to the image recognition hwObject.

8: Repeat steps 5–7 in hwNets of the next layer.

9: A learning/recognition management swObject unifies matching results and controls the weight of the evaluation function.

As the proposed system executes heavy processes with hardware, it can execute complicated processing with software. For example, we can add processing to avoid collisions or to emit sounds of caution after executing tree-structured template matching and recognizing an obstacle. Thus, one advantage of hw/sw complex systems is that it is easy to apply recognition systems to them.

3.4. Feature-Region Learning Unit

This section describes the Feature Region Learning (FRL) circuit (shown in Figure 6) and its implementation on Vocalise.

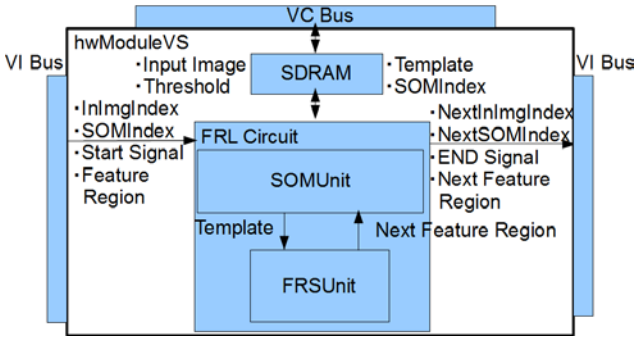

Figure 6. Overview of the FRL circuit

3.4.1. Feature-Region Learning (FRL) Circuit

Feature-region learning is tree-structured processing. The FRL circuit is in charge of processing the tree-structured node unit and consists of the SOM circuit and the feature-region search (FRS) circuit.

First, images are input to the SOM circuit for acquiring templates. Then, the FRS circuit acquires feature regions based on the templates and sends feature regions to nodes in the next layer. Learning narrowed-down feature regions step-by-step in this way, hierarchically from low-

resolution to high-resolution layers, creates a tree-structured feature template.

3.4.2. The SOM Circuit

Figure 7 shows the an overview of the SOM circuit, which is connected to both the FRS circuit and the FPGA Internal Bus (FIB). In-Img Port is a port for reading input images, and RefVec Port is a port for writing reference vectors. The SOM Unit STT carries out overall sequence processing in this architecture. SOM Core takes charge of calculating the distance between an input vector and reference vectors as well as updating reference vectors in the SOM learning. The same number of processing elements (called SOM PEs) as there are reference vectors execute parallel distributed processing.

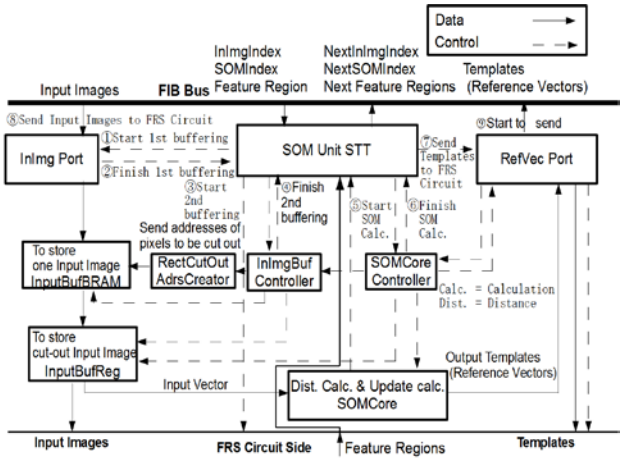

Figure 7. Overview of the SOM circuit and its processing flow

First, loaded images are sequentially written to the block random access memory (BRAM) and distributed RAM; they are then input into SOM Core by the InImg Buf Controller and Rect CutOut Adrs Creator. Reference vectors are written to the host PC's main memory from the buffers in SOM Core after the learning iteration finishes. The SOM circuit that has finished the learning iteration transmits the acquired templates and also inputs images to the FRS circuit. After the FRS circuit finishes calculating feature regions, the feature regions are returned to the SOM circuit. These feature regions, NextIn-ImgIndex, and NextSOMIndex are sent to the next FRL circuit (NextSOMIndex is an index that indicates which SOM (node) in the next layer should be trained using the images that have been learned by the present SOM).

3.4.3. The Feature Region Search (FRS) Circuit

An FRS circuit is a circuit that acquires feature regions. Figure 8 shows an overview of the FRS circuit, which consists of ImgParaIn, FRSCore, and Filter.

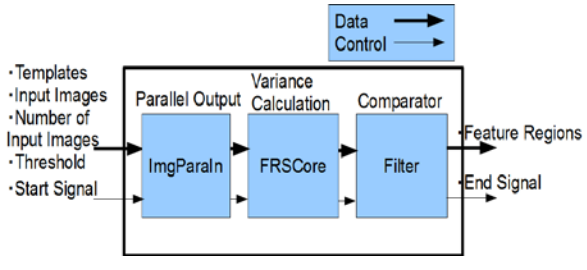

Figure 8. Overview of the FRS circuit

Figure 9 shows the processing flow of the FRS circuit. First, templates that are learned by the SOM circuit are set in a buffer. The pixels of input images sent from the SOM circuit are input to ImgParaIn. ImgParaIn outputs different sub-regions to 17 PEs in FRSCore. Each PE in FRS Core calculates the inner product between a template and an input image and the variance of the inner product values. The calculated variances are passed through Filter to determine high-variance regions (regions where the variance is higher than a certain threshold) and low-variance regions (regions where the variance is lower than a certain threshold), which are then output to the SOM circuit.

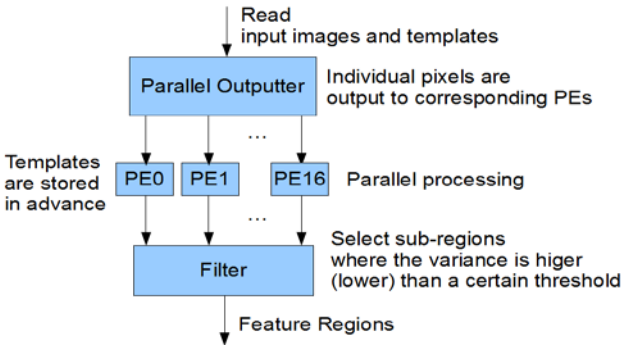

Figure 9. Processing flow of the FRS circuit

3.4.4. Implementation of FRL Circuits on Vocalise

The merit of using Vocalise (which is an FPGA array) is that it can build the most suitable processing elements and placement structure. The proposed system executes four-layer feature-region learning to make four child nodes and 85 nodes. Figure 10 shows the node placement on Vocalise. The address z-y-x in the figure corresponds to the address of Figure 4.

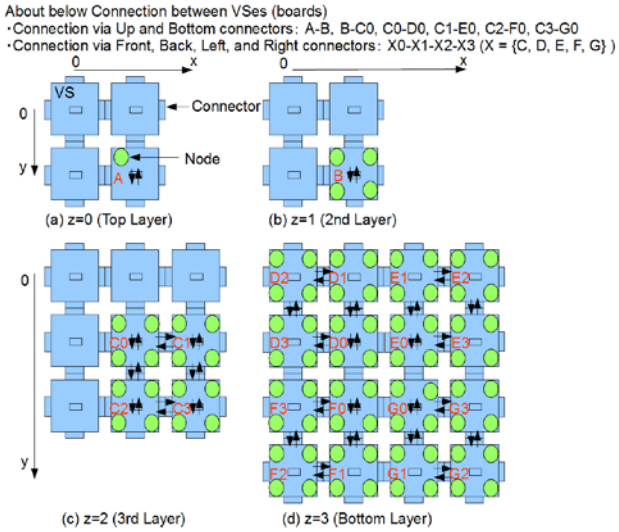

Figure 10. Processing flow of the FRS circuit

We assign the processing of n-th layer to the board of address z=n-1 and utilize Vocalise in the three dimensions. We place one node on a VS in the first layer and four nodes on a VS in the second and subsequent layer. Vocalise Communication (VC) Bus realizes communication between software and a node, and Vocalise Internal (VI) Bus realizes communication between adjacent nodes. In this way, parallel processing in the same layer and pipeline processing in different layers are possible,

increasing the parallel degree to 85 (the total number of nodes).

3.5. Image Recognition Unit

Figure 11 shows an overview of the image recognition unit, which uses a tree-structured feature region template that has been learned. Searching the paths of a tree-structured template determines the optimally matched templates (i.e., the optimal solution). Evaluation of each template is executed by tree-structured nodes (the FRL circuit's distance calculators and comparators are diverted to recognition). The calculated result in each node is weighted; the whole weighting process becomes the evaluation function. Plural optimal paths are found by searching not only the best matched templates but also the 2nd best matched templates (and of course, the 3rd, 4th, and so forth can also be acquired), which allows sufficient feature points.

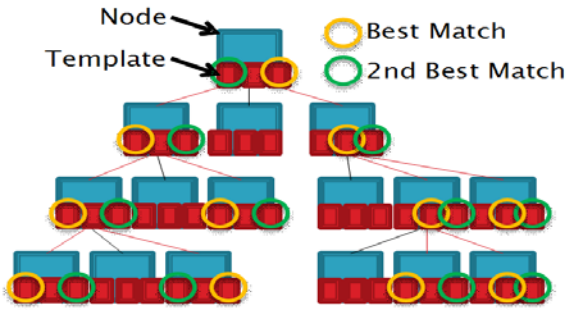

Figure 11. Tree-structured template matching

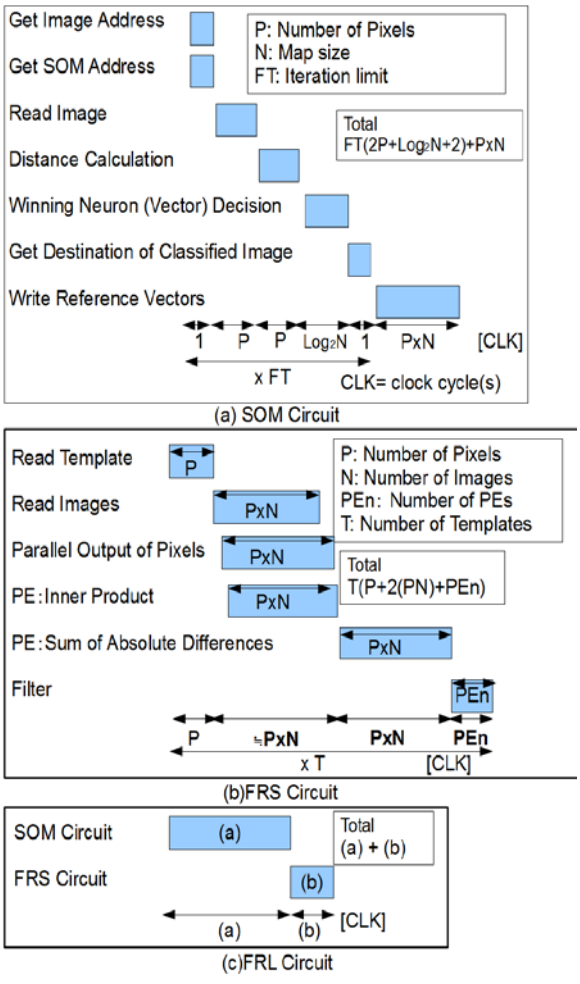

Figure 12. Time charts of the SOM circuit, FRS circuit, and FRL circuit

4. Performance Evaluation

4.1. Performance of the FRL Circuit

Figure 12 shows the time charts of the SOM circuit, FRS circuit, and FRL circuit. The SOM circuit can perform at least 16 times better than sequential processing due to having 16 PEs perform parallel computation of distance calculations and update operations. The FRS circuit can perform at least 17 times better than sequential processing due to having 17 PEs perform parallel computation of 17 sub-regions.

We evaluated the performance of the FRL circuit. Table 1 shows results from the register transfer level (RTL) simulation by Veritak (verilog simulator), from a conventional software-only method, and from actual machine operation by hwModule VL (beginning from the time the software sends a start signal to the circuit to the time the software receives an end signal from the circuit).

Table 1. The FRL Circuit Unit's Processing Speed When Learning

RTL simulation [ms]	Software[ms]	Actual operation[ms]
4.7	3.2×10^2	16

Evaluation conditions will be described when we discuss performance evaluation. One hundred images and 16 pixels (feature region size) from each input image were used; the pixels of an input image were 4 × 4 in the first layer, 8 × 8 in the second layer, 16 × 16 in the third layer, and 32 × 32 in the fourth layer and the resolutions are made by wavelet transform. The SOM's map size was 16, its iteration limit was 2000, the number of PEs was 17, and the number of templates was 4. Few pixels from each image were used because we used images that had been compressed by wavelet transform. In future, we will regard the input region of images as an interest region; this may expand the region for learning.

A compiler of turbo C++ 2006 compiler, Intel Core i7 (4 GHz) CPU (one core), and Windows 7 (32bit) OS were used in the software's processing. In the RTL simulation, we assumed a circuit operation frequency of 66 MHz. In addition, the hwModuleVL used for actual machine operation was the FPGA board that is accessible to the host CPU through a PCI slot. The FRL circuit is supposed to be implemented on the hwModuleVS of Vocalise, but instead hwModuleVL was used for the experiment; its operation frequency was 33 MHz.

The result from the actual machine operation was approximately 3.4 times poorer than the simulation result (shown in Table 1).

The first cause of delay in the actual machine operation was an operation frequency was of 33 MHz (compared to the 66 MHz assumed in the RTL simulation) that occurred due to the critical path in the circuit. The second cause of delay was a communication bottleneck between the hardware and software. We believe the actual processing performance could be closer to simulation results if circuit optimization were conducted.

In addition, the processing performance acquired by simulation was 68 times than that obtained by using a conventional software method. The processing performance acquired by actual machine operation of our proposed system was 20 times better compared to the software method. However, because we believe that machine operation can achieve performance nearer to

simulation results (as stated above), we assume the simulation performance as the processing performance of the FRL circuit unit when we evaluate the performance of the proposed system in the next section.

4.2. Performance of the Feature-Region Learning with Vocalise

Vocalise's SDRAM access, VI Bus, and VC Bus are thought to cause bottlenecks in the proposed method. Figure 13 shows an estimate of the bottlenecks. Note that wait states of (a) – (d) occurs at each VI Bus, VC Bus, and SDRAM of Vocalise. Figure 13 (a) and (b) show VI Bus's communication bottlenecks when nodes send input data to other nodes in the lower layer via a connector. Figure 13(c) shows SDRAM's access bottlenecks when nodes read input data and write results. Figure 13(d) shows VC Bus's communication bottlenecks when a host CPU reads results from SDRAMs on hwModuleVSes.

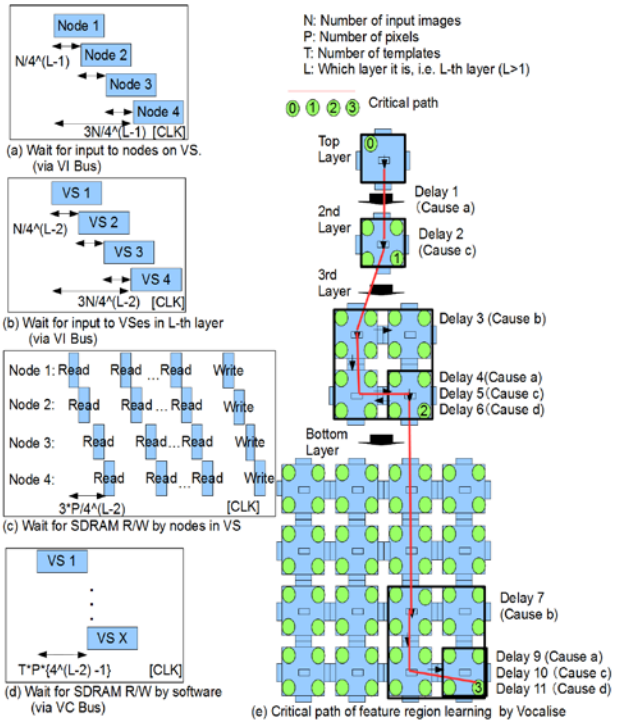

Figure 13. Estimate of bottlenecks in the proposed system

Table 2. Feature-Region Learning Delays (Wait Times) with Vocalise

	The number of clock cycles[clk]	Time [μs]	Cause
Delay 1	225	3.4	a
Delay 2	192	2.9	c
Delay 3	225	3.4	b
Delay 4	57	0.86	a
Delay 5	192	2.9	c
Delay 6	192	2.9	d
Delay 7	225	3.4	b
Delay 8	15	0.23	a
Delay 9	192	2.9	c
Delay 10	960	14	d
Total Delay	2473	37	

Table 2 shows an estimate of the critical path delay using the tree structure shown in Figure 13(e). The total delay is 37 μs . Template transfer to software (d) causes the worst bottleneck, possibly because the number of nodes increases in the lower layers, thus lengthening the wait time. One way to remove a bottleneck is to increase the number of VSes to increase communication links. We believe bottlenecks can be improved depending on how Vocalise is used.

The proposed method's processing speed is shown below. Theoretically, every node processing in the same layer can be processed in parallel. Therefore, the time required for completing the whole process is 4 (the number of layers) × T_{FRL} (the processing time of the FRL circuit unit). On the other hand, in the case of sequential processing done by software, the total processing time is 85 (the number of nodes) × T_{FRL}. Hence, we can estimate the performance of the proposed method as shown in Table 3. The proposed method can increase software's learning speed by 1.4 × 10^3. Thus, we see that the proposed method is able to realize real-time processing, but the software method is not. In addition, after roughly estimating the recognition speeds on the basis of the tree-structured template (of 16 pixels) and input images (of four resolutions: 80 × 60, 160 × 120, 320 × 240, and 640 × 480) when searching for the first and second best matched templates, the proposed system is 13 times faster than the software method.

Table 3. Comparison of Processing Speeds for the Proposed System and a Software-Only Method

	Proposed system[ms]	Software[ms]
Learning	19	2.7× 10^4
Recognition	5.7	78

There are several reasons why the recognition unit results in less performance enhancement than does the feature learning unit.

First, the recognition algorithm (which uses coarse-to-fine template matching) does not require high computational power when compared with the learning algorithm because it searches only narrowed-down regions of input images and does not require much iteration compared to the learning algorithm.

Second, to search regions, the recognition unit has to cut out regions of input images many more times when compared with than the learning unit and doing so is not designed for efficient processing (i.e., in parallel). Of course, the FRS circuit is designed to quickly search regions, but it is specialized for feature-region searches, which means that we cannot disperse this module to the recognition unit; at this time only the FRL circuit's SOM circuit is thus dispersed. With the SOM circuit, inputting cut out regions is sequentially processed, which we believe impedes the system. The recognition unit's processing time could be accelerated if we change the circuit to process search regions in parallel as pipeline processing is often done for block matching.

Third, the recognition unit utilizes fewer resources when compared with SOM. The recognition unit requires 15 nodes when searching for the first and second best matched templates; however, the SOM learning unit requires 85 nodes. However, when more feature points are needed, you can process up to 85 feature points (the number of nodes) can be used simultaneously by the recognition unit because up to 85 nodes can be used for parallel processing. Use of the recognition unit is thought to be much faster than use of software when more feature points are needed.

4.3. Performance of Mobile Systems

The proposed method utilizes 22 hwModuleVSes. As the Xilinx FPGA in the VS has 4 million system gates (55000 slices), it is thought that the proposed method's becomes 88 million system gates (1210000 slices). These numbers suggest that large-scale circuits are indispensable to real-time learning, and, indeed, we see that the robot on which we implement our proposed method requires an enormous circuit scale. We implement the proposed method on Vocalise, which is loaded on our robot as shown in Figure 14. We call the robot Vocalibot. The circuit scales of the whole processes required by Vocalibot are shown in Table 4, which is a partly modified version from the table presented in [14]. The Leg Control and Arm Control circuits generate pulses to control the stepping and servo motors, respectively, installed on Vocalibot. The Stereo Distance circuits preprocess images from cameras to acquire the information required to execute triangulation. The Voice Recognition circuit preprocesses voice data mainly by wavelet transform and executes continuous and pipelined template matching to recognize phonemes (which are the smallest unit of speech). The Voice Synthesis circuit consists of a large number of shift registers and simulates voice waves that propagate in the human vocal tract. The Video Codec circuit compresses video by motion estimation that is realized by block matching circuits with a Single Instruction Multiple Data (SIMD) structure. The Transmission Control Protocol / Internet Protocol (TCP/IP) Stack circuit executes pipeline processing of the TCP/IP processing required for internet communication. The Video Codec circuit and the TCP/IP Stack circuit compose the Internet Booster to accelerate Web applications such as Skype [18].

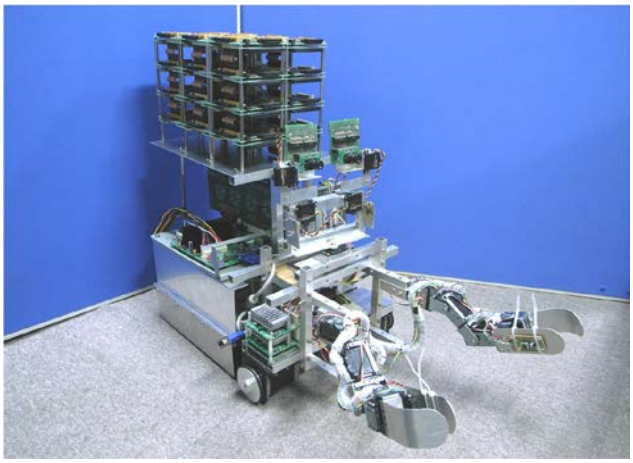

Figure 14. Vocalibot

Table 4. Circuit Scale of Brain Processes for Vocalibot

Brain Process		Size of Circuit (Slices)	
		Current	Expected
Leg Control	Wheel Control	4000	4000
	Arm Control	3000	15000
Image Recognition	Stereo Distance	1200	4000
	Image Learning Recognition	27500	1210000
Voice Recognition		15200	15200
Voice Synthesis		25400	76220
Video Codec		13000	13000
TCP/IP Stack		8300	41500
Total		97600	1378900

Vocalibot will utilize about 30 VSes in total and its total circuit scale is estimated to be 1.4 million slices. A system with circuit-scale scalability like Vocalise is indispensable to a robot, and robots need to be connected to a large number of sensor circuit boards such as camera drivers and motor drivers. We believe that FPGA arrays, which provide more IO connectors than CPU/GPU, are better suited for robots. Some FPGAs have much higher capacities than the ones used in this study, and the proposed algorithm can be implemented on any of them. However, while this allows for a more free pipelined structure (because the inside of the FPGA can be freely programmed), the design becomes flat (i.e., 2D). The 3D FPGA, Vocalise enables us to build circuit structures the same way we would assemble blocks, which saves us the trouble of thinking about 3D algorithms while working in a lower dimension.

Furthermore, even high capacity FPGAs will not provide sufficient logic capacity if the number of layers and PEs increases. Also, when considering robots, we believe that utilizing plural FPGA board is important for securing the number of input/output (IO) connectors, which is why we believe that there is merit in the Vocalise implementation. Building Vocalise with high capacity FPGAs would definitely be even better because, in that case, there would be no need for circuit scale awareness, which is a disadvantage of FPGA implementation. Doing so would also improve development speeds because naive FPGA implementation is possible. However, under current conditions, cost becomes an important factor when choosing which type of FPGA to use because high capacity FPGAs are very expensive.

Power consumption is another important factor to consider. Frequent battery charges for mobile systems is not possible outside, which means that, practically speaking, low-power systems are required. In terms of power consumption, we believe that a system with Vocalise is effective. Such a system's performance per FPGA watt is 25 times better than that of a CPU/GPU system, according to the joint research of Swiss university ETH Zurich and Xilinx [19]. The system's power consumption performed similarly when compared with that of the HOG and Viola–Jones method [5,20].

The power consumption of hwModuleVS and CPU are reported in [21] and shown in Table 5. The operating consumption of 22 VSes in the proposed system is estimated to be 74.8 W when we assume that the FRL circuit's power consumption is the same as that of the Poisson equation circuit. We see that the operating consumption of Vocalise is 2.4 times more that of a single CPU.

Table 5. Power Consumption When Calculating the Poisson Equation

	Standby [W]	Operating [W]
hwModuleVS × 1	2.2	3.4
CPU × 1	-	31.8

In addition to robots, our system can be implemented for use in high-speed mobile systems such as cars. A mobile system running at 100 km/h moves approximately at 30 m/s. It is thought that the proposed system can handle recognition processing, including post-processing, at around a few seconds. In other words, our proposed system demonstrates high performance that an object 30 m

away can be recognized, and post-processing (e.g., collision avoidance) can be completed before accidents occur. Such high performance cannot be achieved without hardware preprocessing.

5. Conclusion

We propose an image recognition system that learns feature regions using the FPGA Vocalise for use with mobile systems, and show that the proposed system can be easily designed at the abstraction level of object-oriented design by using an hw/sw complex system. In addition, we show that reusing the circuit is as easy as application of the recognition system. The results of the RTL simulation of the FRL circuit show that the learning speed of one node in the tree structure is 68 times faster than the learning speed obtained with software. We also show that we can freely connect an FRL circuit to another FRL circuit via VI Bus and that this enables accelerated pipeline processing. The learning speed of the proposed method is estimated to be 1.4×10^3 times faster than that obtained with software, and the recognition speed is 13 times faster. Furthermore, we demonstrate that it is suitable for mobile systems to implement intellectual processing on FPGAs in terms of circuit scale and power consumption.

In the future, we will evaluate the proposed system as it actually operates. In addition, we will conduct combined operations of vehicle control and image learning by loading the proposed system onto the Vocalibot we are still developing. Furthermore, we will conduct experiments to confirm that the proposed system can learn scenic features—in other words, that obstacles in passing scenes may be learned the same way that feature regions such as noses and chins in facial images can be automatically learned through feature-region learning.

Acknowledgement

Portions of the research in this paper use the FERET database of facial images collected under the FERET program, sponsored by the DOD Counterdrug Technology Development Program Office. The authors would like to thank Enago (www.enago.jp) for the English language review.

References

[1] P. Viola and M. Jones, "Robust real-time face detection," in *IEEE International Conference on Computer Vision*, vol.2, p.747, 2001.

[2] Y. Freund and R. E. Schapire, "A decision theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, No. 1, Vol. 55, pp. 119-139, 1997.

[3] J. P. Harvey, "*Gpu acceleration of object classification algorithms using nvidia cuda*," Master's thesis, Rochester Institute of Technology, Rochester, NY, Sept. 2009.

[4] Dalal, N.; Triggs, B., "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol.1, no., pp.886, 2005.

[5] Ma, X.; Najjar, W.A.; Roy-Chowdhury, A.K., "Evaluation and Acceleration of High-Throughput Fixed-Point Object Detection on FPGAs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.PP, no.99, pp.1, 1, 2014.

[6] K. Kudo, Y. Myokan, W. C. Than, S. Akimoto, T. Kanamaru, and M. Sekine, "Hardware object model and its application to the image processing," IEICE Trans. Fund.

[7] M. Yokokawa, I. Sudo, T. Yuno, M. Sekine, "Face detection with the union of hardware and software," IEICE Tech. Rep, Vol.106, No.453, pp.13-18, Jan 2007.

[8] Y. Usami, H. Kotaki, K. Takahashi, M. Sekine, "The voice recognition circuit by using hardware and software complex", IEICE Tech. Rep, EA2007-112, pp.1-6, 2008.

[9] Y. Ogishima, J. Li, S. Yokota, H. Kubo, M. Sekine, "Voice Recognition System using hw/sw Complex," IEICE Tech. Rep, RECONF2014-43, vol.114, no.331, pp.51-56, Nov 2014.

[10] H. Kubo, J. Li, S. Yokota, Y. Ogishima, M. Sekine, "Mobile robot system based on hw/sw Complex System using 3D FPGA-Array System "Vocalise"", IEICE Tech. Rep, RECONF2014-37, vol.114, no.331, pp.19-24, Nov 2014.

[11] M. Sekine, T. Kanamaru, H. Ito, "Multi-level Matching for Detecting Faces", *J. IEICE*, Vol.J86-A, No.9, pp.969-973, Sep 2003.

[12] T. Yuno, I. Sudo, M. Yokokawa, R. Sato, K. Kudo, M. Sekine, "Self-Organizing Map Algorithm that used Base Vector", IEICE Tech. Rep, Vol.106, No.428, pp.1-6, Dec 2006.

[13] M. Ariizumi, B. Ogasawara, H. Tamukoh, M. Sekine, "An Image Recognition System with Hierarchical Feature Learning Function", IEICE Tech. Rep, VLD2011-95, vol.111, no.397, pp.25-30, Jan 2012.

[14] B. Ogasawara, S. Yokota, H. Tamukoh, M. Sekine, "Implementation of an Image Recognition System with Hierarchical Feature Learning Function," IEICE Tech. Rep, RECONF2012-60, vol.112, no.325, pp.77-82, Nov 2012.

[15] S. Yokota, B. Ogasawara, M. Sekine, "A Method for Learning Multi-resolutional Feature Regions", *Workshop on Circuits and Systems 26*, pp.524-529, Jul 2013.

[16] P.J. Phillips, H. Wechsler, J. Huang, P. Rauss, "The FERET database and evaluation procedure for face recognition algorithms, " *Image and Vision Computing J*, Vol. 16, No. 5, pp. 295-306, 1998.

[17] P.J. Phillips, H. Moon, S.A. Rizvi, P.J. Rauss, "The FERET Evaluation Methodology for Face Recognition Algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 22, pp. 1090-1104, 2000.

[18] Hakaru Tamukoh, Kentaro Hanai, Ryosuke Kurogi, Soichiro Matsushita, Masashi Watanabe, Yuichi Kobayashi, and Masatoshi Sekine, "Internet Booster: A Networked Hw/Sw Complex System and Its Application to Hi-Performance WEB Application," *Proc. of World Automation Congress* (WAC2010), 7th International Forum on Multimedia and Image Processing, 6 pages in CD-ROM, Sep., 2010. Kobe.

[19] Xilinx, "SDAccel Development Environment," Available: http://www.xilinx.c tools/sdx/sdaccel.html. [Accessed Dec. 26, 2014].

[20] Hefenbrock, D.; Oberg, J.; Nhat Thanh; Kastner, R.; Baden, S.B., "Accelerating Viola-Jones Face Detection to FPGA-Level Using GPUs, " *18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, vol., no., pp.11,18, 2-4 May 2010.

[21] Y. Atsumari, J. Li, H. Kubo, H. Tamukoh, M. Sekine, "A 3D FPGA-Array HPC System "Vocalise" and its Performance Evaluation, " IEICE Tech. Rep, Vol.112, No.321, pp.201-206, Nov 2012.