Fast FPGA-Based Multi-Object Feature Extraction

Qingyi Gu, Takeshi Takaki, and Idaku Ishii, *Member, IEEE*

***Abstract*—This paper describes a high-frame-rate (HFR) vision system that can extract locations and features of multiple objects in an image at 2000 fps for 512×512 images by implementing a cell-based multi-object feature extraction algorithm as hardware logic on an FPGA-based high-speed vision platform. In the hardware implementation of the algorithm, 25 higher-order local auto-correlation features (HLACs) of 1024 objects in an image can be simultaneously extracted for multi-object recognition by dividing the image into 8×8 cells concurrently with calculation of the 0th and 1st order moments to obtain the sizes and locations of multiple objects. Our developed HFR multi-object extraction system was verified by performing several experiments; tracking for multiple objects rotating at 16 rps, recognition for multiple patterns projected at 1000 fps, and recognition for human gestures with quick finger motion.**

***Index Terms*—high-frame-rate vision, multi-object tracking, image recognition, hardware implementation.**

I. Introduction

MULTI-OBJECT extraction is a fundamental operation in image-based target tracking and pattern recognition. The Rosenfeld algorithm [1] was developed in the 1960s to extract and label connected components in a binary image, and many multi-object extraction methods have been developed for image segmentation based on color, intensity, texture, or motion. Recent research developments in image segmentation for multi-object extraction include watershed transformation [2], [3], [4], graph-cut methods [5], [6], [7], [8], clustering methods [9], [10], [11], neural network approaches [12], [13], [14], [15]. Most of these methods have been verified as robust segmentation methods for complicated scenes that are suitable for many application fields such as multimedia, medical imaging, or remote sensing. However, most of the video cameras used in these systems are restricted to video signal formats (e.g., NTSC 30 fps and PAL 25 fps) that are designed based on the characteristics of the human eye, which means that the processing speed of these systems is limited to the recognition speed of the human eye. However, there is a strong demand for high-speed and real-time multi-object extraction for machine inspection in many application fields, such as the factory automation, biomedical, and robotics fields, where high-speed operations are carried out. These high-speed operations can be tracked and inspected using high-speed vision systems that work as intelligent sensors at hundreds of hertz or more, even when they are difficult to inspect with using human eye.

In order to overcome the restrictions posed by conventional video signals, many machine vision systems that can operate at high frame rates of 1000 fps or more have been developed. Vision chips [16], [17], [18], [19] are one-chip vision systems with resolutions not more than tens of thousands of pixels for high-frame-rate (HFR) video processing that use integrating sensors and processors on a compact die. Field-programmable gate array (FPGA)-based high-speed vision platforms have been developed for the hardware implementation of various types of image processing algorithms such as massively parallel co-processors for multi-target tracking [20], a high-speed Hough transform processor on an FPGA [21], and a high-speed vision platform for real-time video processing of 1024×1024 images at 1000 fps [22]. If we could implement a real-time function to extract image locations and features of multiple objects in an image as hardware logic on such FPGA-based high-speed vision platforms, it would become possible to accelerate multi-object tracking and inspection for various applications at a higher frame rate than the NTSC frame rate of 30 fps.

However, conventional multi-object extraction methods are not always suitable for hardware implementation because most require a large memory area, depending on the image size and iterative processes that are applied to all pixels in an image; it makes it difficult to accelerate multi-object extraction using a high-speed vision platform based on the hardware implementation. In this paper, we focused on multi-object feature extraction based on connected component labeling, and we propose an HFR multi-object extraction system based on the hardware implementation of an improved multi-object feature extraction algorithm using a cell-based labeling algorithm [23]. This method can reduce the computational complexity and memory consumption for the labeling process of multi-object extraction. Connected component labeling is a widely used method in machine vision for the extraction of connected regions in a binary image. Clearly binarized images are required for multi-object extraction based on connected component labeling, but acceleration is required for automated blob inspection in factory automation to extract, track, count, and inspect multiple products and objects that move at high speed under well-designed artificial illumination.

Section II summarizes related works of connected component labeling algorithms to extract multiple objects in a binary image and their problems in hardware-implementing them. Section III describes the cell-based multi-object feature extraction algorithm that can localize multiple labeled objects in a binary image and their higher-order local auto-correlation (HLAC) features [24] as shift-invariant image features for multi-object recognition; the algorithm is so suitably designed for hardware implementation that can reduce computational complexity and memory consumption in labeling process for multi-object extraction. In Section IV, the execution times, accuracies, and memory consumption of our algorithm on a

Q. Gu, T. Takaki, and I. Ishii are with Department of System Cybernetics, Hiroshima University, Higashi-Hiroshima, Hiroshima 739-8527, Japan e-mail: gu@robotics.hiroshima-u.ac.jp (see http://www.robotics.hiroshima-u.ac.jp/).

Manuscript received November 14, 2011
Manuscript revised February 6, 2012

personal computer (PC) are evaluated for several image patterns, compared with several conventional connected component labeling algorithms. Section V provides the outline of the hardware implementation of the cell-based multi-object feature extraction algorithm on a high-speed vision platform and the specifications for performing 2000 fps multi-object extraction on 512×512 images. In order to verify the performance of our HFR multi-object extraction system, several experimental results for high-speed moving objects are given in Section VI.

II. Related works

Many connected component labeling algorithms such as the Rosenfeld algorithm [1] have been proposed to scan twice or more times over an entire image for extracting a label map of connected components in an image. To reduce the number of scanned pixels for labeling, several multi-scan labeling algorithms have been improved such as labeling methods with bi-directional scanning [25], [26] and two-scan run-based labeling [27], [28]. However, most of them still require high memory consumption and two-scan at least, which are disadvantages in accelerating labeling process. One-scan labeling algorithms that can scan connected components in a binary image by tracking their contours, have been proposed [29], [30]. Their computational speeds depend on the complexity of the connected components and they are not suitable for hardware implementation because of their irregular image memory access in tracking the contours of the connected components. Many parallelized labeling algorithms [31], [32], [33] have been developed for implementation of parallel computers; they can label sub-images on divided block regions of a binary image in parallel. Most of them still need to carry out complicated merge processing for the divided block regions and labeled data of the image size have to be stored during processing.

In recent years, several attempts have been also reported for accelerated multi-object extraction by hardware-implementing connected component labeling algorithms on FPGAs. Benkrid et al. developed an FPGA-based connected component labeller with two 4 MB external SRAMs based on multi-scan labeling [34], and connected component labeling for 1024×1024 and 512×512 input images was performed at 68 fps and 278 fps, respectively. Flatt et al. developed an FPGA-based parallelized labeling architecture [35] that can label connected components in an image by scanning the entire image twice in parallel in units of 4 pixels. Whereas external DDR-SRAMs are required for input images and output label images, connected components in a 720×480 image can be extracted at 574 fps in best case, and it may be decelerated because the computation time of merge processing for the divided block regions depends on the complexity of the connected components. These labeling hardwares are not always suitable for real-time processing of larger images at higher frame rates because of the data transfer bottleneck between FPGAs and external memories. Appiah et al. developed an FPGA-based two-scan run-based labeling system [36] that can extract connected components in 1024×1024 and 640×480 images at 74 fps and 253 fps, respectively, by utilizing block RAMs of an FPGA for storing a label map. An FPGA-based object extraction and labeling system [37] was developed that can simultaneously execute background differencing and connected component labeling of 640×480 images at 35 fps. Bailey et al. developed a single-pass labeling algorithm [38] that avoids the need for buffering the input image, while their FPGA implementation was reported to use 640×480 images at 60 fps [39]. Kumar et al. accelerated the single-pass labeling algorithm using a scalable processor on an FPGA platform [40]. However, most of these FPGA-based implementations cannot extract all the connected components when there are thousands of complex-shaped objects in a binary image. This constraint is caused by the limited memory size of the block RAMs on the FPGA, and it is slightly difficult to store image data of 1 Mbytes or more without using any external memory with present integration FPGA technology. Thus, high memory consumption in connected components labeling often makes difficulties in accelerating multi-object extraction for a higher-resolution image by hardware-implementing a connected components labeling algorithm.

III. Cell-Based Multi-Object Feature Extraction

A. Cell-Based Labeling Algorithm

As above-mentioned in the previous section, most of multi-object extraction systems using connected components labeling still require (1) sequential image scan for all pixels in an image with computational complexities of the order or image size, which make difficulties to parallelize labeling process, (2) many label equivalences for connected components when there exist non-convex objects and isolated small objects in an image, which often require large memories of image size or more to store image features of the labeled regions. To reduce such computational complexities and memory consumption in connected components labeling, Gu et al. have proposed a cell-based labeling algorithm [23] to extract the 0th and 1st order moments of multiple objects in a binary image; this can reduce the number of scanned pixels for labeling and memory size to store label equivalences without accuracy degradation in space resolution by dividing an image into subimage regions of a certain size as cells.

The cell-based labeling algorithm can obtain moments $M_{pq}(O_l)$ for labeled regions O_l $(l = 0, \ldots, L-1)$ in a binary image $B(x, y)$ of $N \times N$ pixels by dividing $B(x, y)$ into M^2 cells Γ_{ab} $(a = 0, \ldots, M-1, b = 0, \ldots, M-1)$ of $n \times n$ pixels when $N = nM$,

$$\Gamma_{ab} = \{(x, y) | (an + s, bn + t),\ 0 \le s < n, 0 \le t < n\}. \quad (1)$$

The 0th and 1st moments $M_{00}(O_l)$, $M_{10}(O_l)$, and $M_{01}(O_l)$ in a labeled region O_l are described as,

$$M_{pq}(O_l) = \sum_{(x,y) \in O_l} x^p y^q B(x, y), ((p,q) = (0,0), (1,0), (0,1)). \quad (2)$$

In the cell-based labeling algorithm, its computational complexity and memory consumption for labeling process can be reduced by exchanging its computational sequence in labeling process of divided cells after calculating the moments for cells of $n \times n$ pixels based on the additivity in moment calculation;

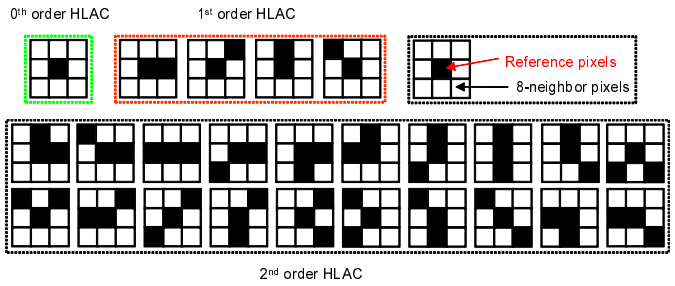


Fig. 1. 25 local patterns for HLACs.

moments $M_{pq}(\Gamma)$ in a region Γ can be calculated by adding moments $M_{pq}(\Gamma_k)$ for its sub-regions Γ_k $(k = 0, \ldots, K-1)$ as follows,

$$M_{pq}(\Gamma) = \sum_{k=1}^{K} M_{pq}(\Gamma_k), \tag{3}$$

where the sub-regions Γ_k satisfy the following conditions,

$$\Gamma = \sum_{k=1}^{K} \Gamma_k, \quad \Gamma_i \cap \Gamma_j = \emptyset \quad (i \neq j). \tag{4}$$

The results labeled with the cell-based labeling algorithm are not perfectly matched with those with pixel-based connected component labeling algorithms when $n > 1$, because there are cases that the pixels belonging to different connected components in the same cell or neighboring cells are identified as the same regions based on the connectivity of cells. Although we have to consider a trade-off relationship between its computational complexity and equivalency to pixel-based connected component labeling, the computational complexity and memory consumption for labeling process in the cell-based labeling algorithm can be reduced in the cell-level complexity of the order $O(M^2)$ when $n > 1$, that is, $1/n^2$ of the pixel-level complexity of the order $O(N^2)$ in pixel-based connected components labeling algorithms. Thus, the cell-based labeling algorithm is suitable for hardware implementation on a high-speed vision platform to extract multiple objects in a binary image at a higher frame rate.

B. HLACs and Their Additivities

HLACs [24] are shift-invariant features that can be used for pattern recognition; the n-th order HLAC in a region Γ is calculated by the following higher order auto-correlation between a reference pixel and its neighbors:

$$\sum_{\boldsymbol{r} \in \Gamma} I(\boldsymbol{r}) I(\boldsymbol{r} + \boldsymbol{a}_1) \cdots I(\boldsymbol{r} + \boldsymbol{a}_n), \tag{5}$$

where $I(\boldsymbol{r})$ is the brightness at a reference pixel $\boldsymbol{r} = (x, y)$ and $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ represent displacements. Corresponding to the size of a neighbor area, we have to store the image data in multiple rows when the HLACs are calculated using raster image scanning. To reduce the memory consumption of image data storage from previous lines as much as possible, we assume that the displacements $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are limited in a neighbor area of 3×3 pixels around $\boldsymbol{r}$, and the 0th, 1st, and 2nd order HLACs are calculated for a region Γ in a binarized image $B(x, y)$. Note that 3×3 pixels is the minimum neighbor area that can be used to calculate the 0th, 1st, and 2nd order HLACs for curvature information. The number of HLACs to be calculated is 25 (0th order: 1, 1st order: 4, and 2nd order: 20), and the 25 local patterns of 3×3 pixel images, whose center pixel is the reference pixel, are shown in Fig. 1. The 25 HLACs $F_i(\Gamma)$ $(i = 0, \ldots, 24)$ are expressed as follows:

$$F_0(\Gamma) = \sum_{(x,y) \in \Gamma} B(x, y), \tag{6}$$

$$F_1(\Gamma) = \sum_{(x,y) \in \Gamma} B(x, y) B(x+1, y), \tag{7}$$

$$F_2(\Gamma) = \sum_{(x,y) \in \Gamma} B(x, y) B(x+1, y-1), \tag{8}$$

$$\vdots$$

$$F_{24}(\Gamma) = \sum_{(x,y) \in \Gamma} B(x, y) B(x-1, y-1) B(x+1, y-1). \tag{9}$$

In the same way as the moments in Eq. (3), these HLACs $F_i(\Gamma)$ can be calculated by adding HLACs for its sub-regions Γ_k that satisfy the condition in Eq. (4),

$$F_i(\Gamma) = \sum_{k=1}^{K} F_i(\Gamma_k). \tag{10}$$

This additivity implies that the HLACs for an image region can be obtained by calculating the HLACs for its divided cells and combining them as well as the moments in the cell-based labeling algorithm. This commutativity in HLAC calculation enables to expand the cell-based labeling algorithm for shape-based multi-object recognition using HLACs easily.

C. Cell-Based Multi-Object Feature Extraction Algorithm

The cell-based labeling algorithm in [23] only concerns the 0th and 1st order moments to obtain sizes and locations of labeled objects in a binary image for multi-object tracking; these moments are not always sufficient for shape-based blob analysis to identify multiple complex-shaped objects in a binary image. In this paper, we expand the cell-based labeling algorithm as a cell-based multi-object feature extraction algorithm that can obtain higher-dimensional shift-invariant features of labeled objects in an image as well as their 0th and 1st order moments.

Our cell-based multi-object feature extraction algorithm can obtain 25 HLACs $F_i(O_l)$ $(i = 0, \ldots, 24)$ for labeled regions O_l in a binary image $B(x, y)$ as follows,

$$F_i(O_l) = \sum_{\boldsymbol{r} \in O_l} B(\boldsymbol{r}) B(\boldsymbol{r} + \boldsymbol{a}_1) B(\boldsymbol{r} + \boldsymbol{a}_2), \tag{11}$$

as well as their moments $M_{00}(O_l)$, $M_{10}(O_l)$, and $M_{01}(O_l)$, by dividing of $B(x, y)$ into M cells of Γ_{ab} of $n \times n$ pixels as described in Eq. (4). The 0th and 1st order HLACs are calculated using Eq. (11) with $\boldsymbol{a}_1 = \boldsymbol{a}_2 = \boldsymbol{0}$ and $\boldsymbol{a}_2 = \boldsymbol{0}$, respectively. The algorithm has two processes as shown in Fig. 2; (1) cell-level calculation of HLACs $F_i(\Gamma_{ab})$ and moments $M_{pq}(\Gamma_{ab})$ for M^2 cells, which requires pixel-level computation of the order $O(N^2)$, and (2) object-level calculation for labeling M^2

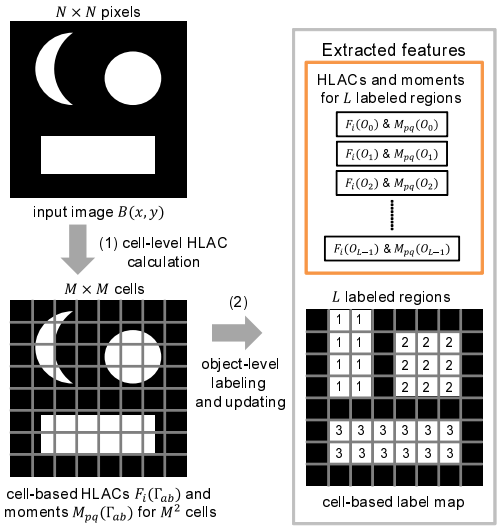


Fig. 2. Cell-based multi-object feature extraction algorithm using HLACs.

cells with HLACs $F_i(O_l)$ and moments $M_{pq}(O_l)$ for L labeled regions O_l, which requires cell-level computation of the order $O(M^2)$.

(1) Cell-level HLAC calculation

The cell-based 25 HLACs $F_i(\Gamma_{ab})$ and the cell-based 0th and 1st order moments $M'_{pq}(\Gamma_{ab})$ are calculated for every cell Γ_{ab} of $n \times n$ pixels:

$$F_i(\Gamma_{ab}) = \sum_{\boldsymbol{r}\in\Gamma_{ab}} B(\boldsymbol{r})B(\boldsymbol{r}+\boldsymbol{a}_1)B(\boldsymbol{r}+\boldsymbol{a}_2), \tag{12}$$

$$M'_{pq}(\Gamma_{ab}) = \sum_{s=0}^{n-1}\sum_{t=0}^{n-1} s^p t^q B(an+s, bn+t). \tag{13}$$

Here, $M'_{pq}(\Gamma_{ab})$ is used to reduce the memory consumption, which is similar to $M_{pq}(\Gamma_{ab})$ using a coefficient $s^p t^q$ with a smaller bit length than that of $(an+s)^p(bn+t)^q$. $M_{pq}(\Gamma_{ab})$ and $M'_{pq}(\Gamma_{ab})$ have the following relationship,

$$M_{00}(\Gamma_{ab}) = M'_{00}(\Gamma_{ab}), \tag{14}$$

$$M_{10}(\Gamma_{ab}) = M'_{10}(\Gamma_{ab}) + an \cdot M'_{00}(\Gamma_{ab}), \tag{15}$$

$$M_{01}(\Gamma_{ab}) = M'_{01}(\Gamma_{ab}) + bn \cdot M'_{00}(\Gamma_{ab}). \tag{16}$$

(2) Object-level labeling and updating

All the cells are labeled by the cell-based 0th order moments $M'_{00}(\Gamma_{ab})$ to extract the HLACs and moments of the labeled regions as connected components in an image. $M'_{00}(\Gamma_{ab})$ is the number of active pixels in Γ_{ab}. The cells are labeled by scanning P_{ab} of $M \times M$ cells from the upper left cell to the lower right cell. P_{ab} is a flag map of $M \times M$ cells to judge whether a cell Γ_{ab} should be labeled or not. This flag map is defined by checking $M'_{00}(\Gamma_{ab})$ with a threshold θ as follows:

$$P_{ab} = \begin{cases} 1 & (M'_{00}(\Gamma_{ab}) \geq \theta), \\ 0 & (\text{otherwise}). \end{cases} \tag{17}$$

When Γ_{ab} is a current cell, we assume that the labels $l_{0b-1}, \ldots, l_{M-1b-1}$ for M cells $\Gamma_{0b-1}, \ldots, \Gamma_{M-1b-1}$ in the previous row $(b-1)$ and the tentative labels $l'_{0b}, \ldots, l'_{a-1b}$ for a cells $\Gamma_{0b}, \ldots, \Gamma_{a-1b}$ in the current row b are already given. For L labeled regions O_l $(l = 0, \ldots, L-1)$, $f_i(l)$ $(i = 0, \ldots, 24)$ and $m_{pq}(l)$ $((p,q) = (0,0),(1,0),(0,1))$ are used as memories to update the HLACs and moments for labeled regions in sequential scanning. The memory $r(l)$ is also prepared for the relabeling process after scanning every row.

The labeling process of cells includes a) a tentative labeling subprocess with updated HLACs and moments in every cell and b) a relabeling subprocess in every row.

a) Tentative labeling sub-process in every cell

When $P_{ab} = 1$, that is, a current cell Γ_{ab} is active, its label is tentatively obtained by using the relationship between its upper cell Γ_{ab-1} and left cell Γ_{a-1b}. To determine a tentative label l'_{ab} and update HLACs $f_i(l'_{ab})$ and moments $m_{pq}(l'_{ab})$, we consider the following five cases:

[Case 1] $P_{ab-1} = P_{a-1b} = 0$
(A) $l'_{ab} = l_{new}$.
(B) $m_{pq}(l'_{ab}) = M_{pq}(\Gamma_{ab})$.
(C) $f_i(l'_{ab}) = F_i(\Gamma_{ab})$.

[Case 2] $P_{ab-1} = 1, P_{a-1b} = 0$
(A) $l'_{ab} = l_{ab-1}$.
(B) $m_{pq}(l'_{ab}) = \hat{m}_{pq}(l'_{ab}) + M_{pq}(\Gamma_{ab})$.
(C) $f_i(l'_{ab}) = \hat{f}_i(l'_{ab}) + F_i(\Gamma_{ab})$.

[Case 3] $P_{ab-1} = 0, P_{a-1b} = 1$
(A) $l'_{ab} = l_{a-1b}$.
(B) $m_{pq}(l'_{ab}) = \hat{m}_{pq}(l'_{ab}) + M_{pq}(\Gamma_{ab})$.
(C) $f_i(l'_{ab}) = \hat{f}_i(l'_{ab}) + F_i(\Gamma_{ab})$.

[Case 4] $P_{ab-1} = P_{a-1b} = 1, l_{ab-1} = l'_{a-1b}$
(A) $l'_{ab} = l'_{a-1b}$.
(B) $m_{pq}(l'_{ab}) = \hat{m}_{pq}(l'_{ab}) + M_{pq}(\Gamma_{ab})$.
(C) $f_i(l'_{ab}) = \hat{f}_i(l'_{ab}) + F_i(\Gamma_{ab})$.

[Case 5] $P_{ab-1} = P_{a-1b} = 1, l_{ab-1} \neq l'_{a-1b}$
(A) $l'_{ab} = \min(l_{ab-1}, l'_{a-1b})$.
(B) $m_{pq}(l'_{ab}) = \hat{m}_{pq}(l_{ab-1}) + \hat{m}_{pq}(l'_{a-1b}) + M_{pq}(\Gamma_{ab})$.
(C) $f_i(l'_{ab}) = \hat{f}_i(l_{ab-1}) + \hat{f}_i(l'_{a-1b}) + F_i(\Gamma_{ab})$.
(D) $r(\max(l_{ab-1}, l'_{a-1b})) = l'_{ab}$.

where l_{new} indicates a new label, and $\hat{f}_i(l)$ and $\hat{m}_{pq}(l)$ are defined as $f_i(l)$ and $m_{pq}(l)$ in the previous cell in scanning, respectively. In sub-process (1), $\hat{f}_p(l)$ and $\hat{m}_{pq}(l)$ are initially set to zero in the left upper cell Γ_{00}. When there exist no upper or left cells around the cells Γ_{0j} or Γ_{i0} $(i = 0, \ldots, M-1, j = 0, \ldots, M-1)$ in the first row or column, the neighboring cells are virtually set as nonactive cells in this paper.

b) Relabeling sub-process in every row

After all the cells in the current row are scanned for sub-process a), their tentative labels l'_{ab} still remain to be updated while the HLACs $f_p(l'_{ab})$ and moments $m_{pq}(l'_{ab})$ for labeled regions have already been updated. To memorize the labels for all the cells in the current row as upper cell data before starting to label cells in the next row, tentative labels l'_{ib} are updated to new labels l_{ib} for all the cells Γ_{ib} $(i = 0, \ldots, M-1)$ in the current row b from right to left when a row is completely scanned. By using the memory $r(l)$, which indicates which labels are unified, the relabeling process is executed as follows:

$$\begin{array}{c} r(l'_{ib}) = r(r(l'_{ib})), \\ l_{ib} = r(l'_{ib}) \end{array} \qquad (i = 0, \ldots, M-1). \tag{18}$$

To reduce the memory areas for the abandoned tentative labels when Case 5 occurs, they are released for recycling as new labels after the relabeling process. The memory $r(l)$ for relabeling is then reset as follows:

$$r(l) = l \quad (l = 0, \ldots, L-1). \tag{19}$$

After scanning all the rows for sub-processes a) and b), the 25 HLACs $F_i(O_l)$, the 0th order moments $M_{00}(O_l)$, and the 1st order moments $M_{10}(O_l)$ and $M_{01}(O_l)$ for labeled regions O_l are finally obtained as follows:

$$F_i(O_l) = f_i(l), \tag{20}$$

$$M_{pq}(O_l) = m_{pq}(l). \tag{21}$$

Thus, the cell-based labeling algorithm is expanded for calculating HLACs of labeled objects in an image as well as calculation of their moments; it enables shape-based multi-object recognition using HLACs without remarkably increasing the computational complexity in labeling process only by introducing cell-based HLAC calculation with a computational complexity of the order $O(N^2)$ in Eq. (12) and updating of HLACs in tentative labeling with a computational complexity of the order $O(M^2)$ based on the commutativity in HLAC calculation.

IV. Evaluation

To emulate the evaluation of the cell-based labeling algorithm in [23], the execution times, accuracies, and memory consumption of our cell-based multi-object feature extraction algorithm were evaluated on a PC by calculating 25 HLACs and moments for each labeled object, compared with those obtained using several connected component labeling algorithms.

A. Execution time on the PC

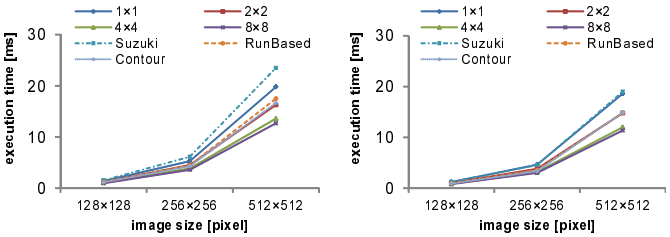
We evaluated execution time using our cell-based multi-object feature extraction algorithm with different cell sizes where the program was implemented on a PC. $N \times N$ images ($N = 128, 256$, and 512) were evaluated for the binary image patterns shown in Fig. 3. These were collected from the image database at the University of Southern California[1], i.e., (a) mandrill and (b) Lena were the 512×512 images that we evaluated, which were binarized using the Otsu method [41]. The cell sizes were set at $n \times n$ ($n = 1, 2, 4$, and 8), while the maximum number of labeled regions was set as half of the cells, $L = M^2/2$. A threshold θ that determined the active cells was set at 1. We used a PC with an ASUSTek P6T7 mainboard, 3.33 GHz CPU, 6GB memory, and a Windows XP Professional 32-bit OS.

Fig. 4 shows the execution times of our cell-based multi-object feature extraction algorithm with different image sizes, which were compared with those from conventional connected component labeling algorithms. All of the conventional labeling algorithms were implemented using object feature extraction for two 1st order moments and 25 HLACs of multiple objects in an image. In the figures, "Suzuki", "RunBased", and

[1]http://sipi.usc.edu/database/

(a) mandrill (b) Lena

Fig. 3. Image patterns in evaluating execution times on a PC.



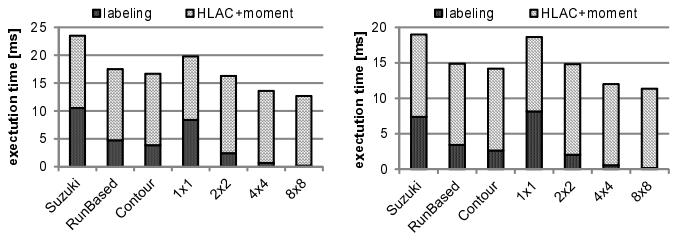
(a) mandrill (b) Lena

Fig. 4. Execution times for our cell-based multi-object feature extraction algorithm and conventional connected components labeling algorithms.

"Contour" show the execution times for connected component labeling. We used 25 HLACs and moments calculation for the labeled regions with Suzuki's four-scan labeling algorithm [26], He's raster-scan-based labeling algorithm [27], and Chang's contour-based one-scan labeling algorithm [30], respectively. The labels "1×1", "2×2", "4×4", and "8×8" show the execution times when $n = 1, 2, 4$, and 8, respectively, when using our cell-based multi-object feature extraction algorithm. For the "mandrill" image containing 512×512 pixels, the execution times with our algorithm when $n = 1, 2, 4$, and 8 were 19.8 ms, 16.3 ms, 13.6 ms, and 12.7 ms, respectively, while those with Suzuki's algorithm, He's algorithm, and Chang's algorithm were 23.5 ms, 17.5 ms, and 16.7 ms. For the "Lena" image containing 512×512 pixels, the execution times when using our algorithm with $n = 1, 2, 4$, and 8 were 18.6 ms, 14.8 ms, 12.0 ms, and 11.4 ms, respectively, while those with Suzuki's algorithm, He's algorithm, and Chang's algorithm were 19.0 ms, 14.9 ms, and 14.2 ms.

For the 512×512 image patterns, Fig. 5 shows the execution time breakdowns for connected component labeling and the HLAC calculation for 25 HLACs and two 1st order moments. When the "mandrill" image was labeled using our cell-based multi-object feature extraction algorithm when $n = 1$ and 8, the execution times for the labeling of cells were 8.4 ms and 0.2 ms, respectively, while those for the HLAC calculation were 10.5 ms and 11.2 ms, respectively. When the same image pattern was labeled with Suzuki's algorithm, He's algorithm, and Chang's algorithm, the execution times for pixel-based labeling were 10.5 ms, 4.7 ms, and 3.8 ms, respectively, while those for the HLAC calculation were approximately 12.9 ms as shown in Fig. 5(a).

The execution time for labeling was significantly decreased with our algorithm when the cell size n increased, whereas the execution time for the HLAC calculations varied little with n and it was similar to that of conventional labeling algorithms. This indicates that our algorithm can be accelerated for the labeling of cells by setting a large cell size, while multi-object



(a) mandrill (b) Lena

Fig. 5. Relationship between execution times and cell sizes.

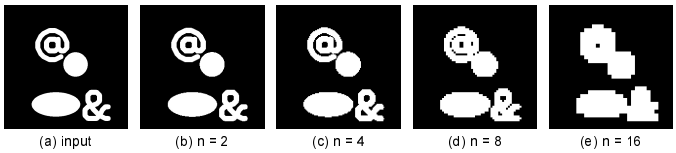


Fig. 6. Flag maps of active cells for labeling.

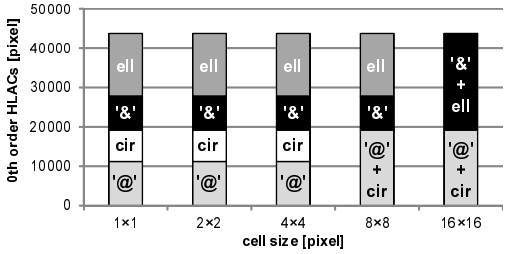


Fig. 7. 0th order HLACs of labeled regions ("@" = the letter "@", "&" = the letter "&", ell = ellipse, cir = circle).

feature extraction can be executed at a much higher speed if sufficient acceleration is provided for HLAC calculation that requires a pixel-level computation that can be easily implemented using the hardware.

B. Accuracy Verification

We evaluated how the labeled results in our cell-based multi-object feature extraction algorithm changed with the cell size n. The 512×512 binary image shown in Fig. 6(a), containing the symbols "@" and "&", an ellipse, and a circle, was labeled using our algorithm with cell sizes of $n = 1, 2, 4, 8$, and 16, while the threshold θ was set to 1. Fig. 6(b)–(e) show the flag maps of the active cells when $n = 2, 4, 8$, and 16. The flag map when $n = 1$ corresponded to the image is shown in Fig. 6(a). The results labeled with our algorithm matched those with conventional labeling algorithms when $n = 1$. In the labeled results, $4, 4, 4, 3$, and 2 labeled regions were counted when $n = 1, 2, 4, 8$, and 16, respectively. This was because the symbol "@" and the neighboring circle were unified in the same labeled region when $n = 8$, while the symbol "&" and the neighboring ellipse were unified when $n = 16$.

Fig. 7 shows bargraphs of the cell-based 0th order HLACs when $n = 1, 2, 4, 8$, and 16, where the 0th order HLAC indicates the number of pixels in the labeled region, which corresponds to the 0th order moment. When $n = 2$ and 4, there were no neighbor connected components and all the 0th order HLACs in the labeled regions exactly matched those when $n = 1$, which corresponded with those calculated using other pixel-based connected component labeling algorithms. When $n = 8$ and 16 with the unification of the neighbor connected components, all the 0th order HLACs exactly matched the summed values from the two unified connected components when $n = 1$. A similar trend was found when calculating the other HLACs and moments for the labeled regions. This indicates that the separability of connected components is determined by the cell size n in our algorithm, while the moments and HLACs are preserved by setting the threshold $\theta = 1$ when there are no unified labeled regions in an image. Our algorithm was not equivalent to conventional pixel-level labeling algorithms, because the same label was often given to different but closely connected components as our algorithm only considered the cell-level connectivity. In our algorithm, there is a trade-off between robustness to small and narrow connected components in an image and the accuracy of the HLAC calculations, which is based by the threshold θ that determines the active cells. This trade-off in cell-based labeling was evaluated in [23].

TABLE I
NUMBERS AND BIT WIDTHS OF LABELS FOR A 512×512 IMAGE.

cell size n	number $L = M^2/2$	bit width $w = \lceil \log_2 L \rceil$
1	131072	17 bits
2	32768	15 bits
4	8192	13 bits
8	2048	11 bits

TABLE II
MEMORY CONSUMPTION FOR A 512×512 IMAGE.

cell size n	(1) QL	(2) Mw	(3) Lw	total
1	8,634,368	1,088	278,528	8,913,984
2	2,158,592	480	61,440	2,220,512
4	539,648	208	13,312	553,168
8	134,912	88	2,816	137,816

unit: byte = 8 bits

C. Memory Consumption

We considered the memory consumption for 25 HLACs and two 1st order moments when executing our cell-based multi-object feature extraction algorithm with a variable cell size n and a variable maximum number of labels L. For an $N \times N$ binary image, the HLAC consumes $(\lceil \log_2 N^2 \rceil + 1)$ bits, and the 1st order moment consumes $\left\lceil \log_2 \frac{N(N-1)(N-2)}{2} \right\rceil$ bits. When T HLACs and two 1st order moments are calculated, they consumed

$$Q = T(\lceil \log_2 N^2 \rceil + 1) + 2 \left\lceil \log_2 \frac{N(N-1)(N-2)}{2} \right\rceil, \quad (22)$$

where $\lceil x \rceil$ denotes the ceiling function that gives the smallest integer $\geq x$. When there are L connected components in the image, the number of bits needed to express their label numbers is $w = \lceil \log_2 L \rceil$ bits. In our algorithm, (1) QL bits for HLACs and moments of L labeled regions, (2) Mw bits for the label numbers of M cells, and (3) Lw bits for the tentative memories for L label numbers in the relabeling process, were required for $M \times M$ cells containing $n \times n$ pixels. The maximum number of labeled regions was set to $L = M^2/2$. Table I shows L and w for a 512×512 image

TABLE III
MEMORY CONSUMPTION IN DIFFERENT IMAGE SIZES AND CELL SIZES.

	cell size				
image size	$n=1$	$n=2$	$n=4$	$n=8$	$n=16$
128×128	429	107	27	7	2
256×256	1,953	487	122	31	8
512×512	8,706	2,169	541	135	34
1024×1024	38,659	9,634	2,401	599	150
2048×2048	168,966	42,115	10,498	2,617	653

unit: kbyte

TABLE IV
MEMORY CONSUMPTION OF A 512×512 IMAGE IN DIFFERENT NUMBER OF HLACS AND CELL SIZES.

total	cell size				
HLACs	$n=1$	$n=2$	$n=4$	$n=8$	$n=16$
10	4,146	1,029	256	64	16
25	8,706	2,169	541	135	34
40	13,266	3,309	826	206	52
55	17,826	4,449	1,111	278	70
70	22,386	5,589	1,396	349	87

unit: kbyte

($N = 512$), while Table II shows the memory consumption when 25 HLACs ($T = 25$) were calculated for a 512×512 image with cell sizes of $n = 1, 2, 4$, and 8. The total memory consumption was 2,220,512 bytes, 553,168 bytes, and 137,816 bytes, respectively, when $n = 2, 4$, and 8. These values were 0.249 times, 0.062 times, and 0.015 times that of 8,913,984 bytes when $n = 1$. This shows that our algorithm can greatly reduce the memory consumption during multi-object feature extraction with a larger cell size.

Table III shows the total memory consumption when executing our algorithm with different cell sizes to calculate 25 HLACs ($T = 25$) and two 1st order moments for different image sizes. The memory consumption increased as the image size N became larger, and there were similar trends where the total memory consumption was significantly reduced with a larger cell size n. For example, we considered the memory consumption using hardware to implement our algorithm in a commercial FPGA (Xilinx XC3S5000-4FG900). This FPGA was designed for high-volume and cost-sensitive consumer electronic applications and it had a block RAM of 239 kbytes. In Table III, the underlined numbers indicate that the memory consumption was less than 239 kbytes. When the block RAM was used completely for hardware implementation, with images of 128×128 pixels, 256×256 pixels, 512×512 pixels, and 1024×1024 pixels, the minimum cell sizes that facilitated the hardware implementation of our algorithm were $n = 2$, 4, 8, and 16, respectively, while we were unable to implement pixel-level connected component labeling ($n = 1$) for a 128×128 image in the FPGA.

Table IV shows the total memory consumption when executing our algorithm with different cell sizes for different number of HLACs when a 512×512 image was processed. The memory consumption increased as the number of HLACs increased, and there was a similar tendency where the total memory consumption was reduced with a larger cell size n. Due to memory restrictions of the FPGA (Xilinx XC3S5000-4FG900), it was unable to implement our algorithm on the FPGA with cell sizes of $n = 1, 2$, and 4 when calculating ≥10 HLACs. With a cell size of $n = 8$, our algorithm can be implemented in the FPGA to calculate ≥40 HLACs. Some block RAMs would be consumed by image data cashing and outputting the data to an external system during the actual implementation on the FPGA.

Based on these results, we verified that we can implement our algorithm with a larger cell size without consuming large amounts of memory resources. This will facilitate the hardware implementation of our high-speed multi-object feature extraction method using an FPGA with limited memory resources. Thus, we can realize real-time multi-object feature extraction at a higher frame rate for larger images by hardware-implementing our cell-based approach on an FPGA.

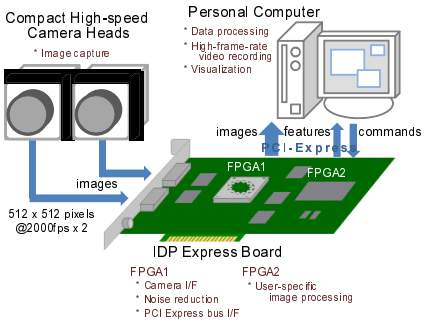


Fig. 8. Configuration of high-speed vision: IDP Express.

V. IMPLEMENTATION

A. High-speed Vision Platform: IDP Express

Our cell-based multi-object feature extraction algorithm was hardware-implemented on a high-speed vision platform, IDP Express [42], and 1024 objects recognition for 512×512 images was simultaneously carried out at 2000 fps. Fig. 8 shows the configuration of this platform, which consists of two compactly designed high-speed camera heads, a dedicated FPGA board (IDP Express board), and a PC.

For the camera head, FASTCAM MH4-10K (Photron) was adopted; its size and weight are 35 mm×35 mm×35 mm and 300 g, respectively. Color or gray 8-bit images were transferred at 2000 fps for 512×512 pixels with digital serial communication in parallel with 6 pixels. Here 8-bit color images can be captured on its image sensor with a Bayer color filter. The IDP Express board was designed as a dedicated FPGA board for high-speed video processing and recording of two 512×512 images transferred at rates as high as 2000 fps. Fig. 9 shows its function block diagram. The board consists of two FPGAs, configuration PROMs for the FPGAs, serial-to-parallel converters for camera inputs, and FIFO (First-In First-Out) memories for data transfer between the FPGAs. One FPGA (Xilinx XC4VFX60, hereafter called FPGA1) is used for camera I/O and PCI-e bus controls, and one FPGA (Xilinx XC3S5000-4FGG900, hereafter called FPGA2) is used for hardware implementation of user-specified functions. This board is connected to PC through 8 lanes of PCI-e buses; 8-bit 512×512 images can be transferred at 4000 fps for two camera heads. The path for video processing and transfer in the IDP Express board is summarized as follows:

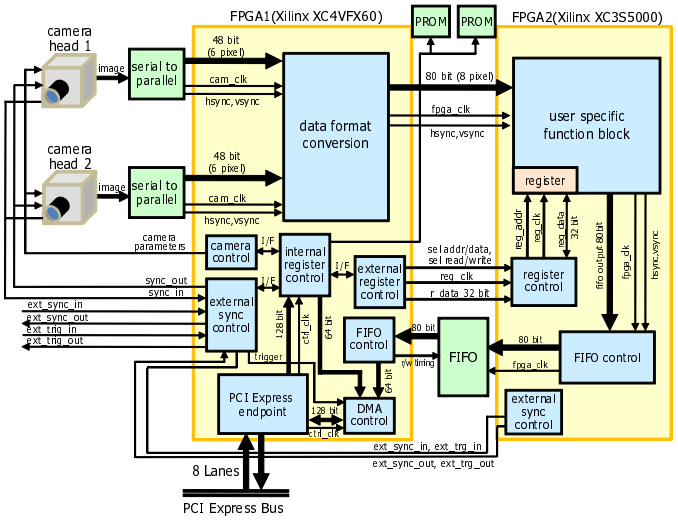


Fig. 9. Function block diagram of IDP Express board.

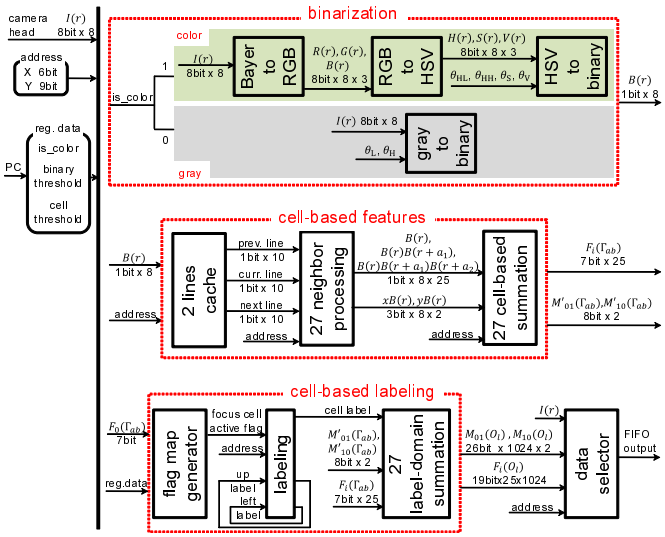


Fig. 10. Schematic data flow of implemented circuit.

(1) Converting image data from serial to parallel

Serial image data from a camera head are converted into 48-bit parallel data (8 bits/pixel) with 6 pixels as a block at the camera head's clock speed of 100.8 MHz. These conversions are executed for the two camera heads in parallel.

(2) Converting data format for image processing

On FPGA1, the 48-bit parallel data for 6 pixels at 100.8 MHz are converted into 40-bit parallel data for 4 pixels at the FPGA2's clock speed of 151.2 MHz to synchronize with the timings for rows and columns of a 512×512 image. The conversions are executed in parallel for two camera heads, and 96-bit parallel data are converted into 80-bit parallel data for 8 pixels at 151.2 MHz.

(3) Executing user-specified image processing

The 80-bit data from FPGA1 are processed in a processing block in FPGA2, where a user-implemented image processing hardware algorithm is executed in real time for the two camera head inputs. The processing block can use register values for parameters in the algorithm; the register values are updated at a clock speed of 50.4 MHz on commands from a PC.

(4) Transferring image data and processing results to the PC

Both the input images and processed results are transmitted through FIFO memories from FPGA2 to FPGA1 in parallel with 80 bits at 151.2 MHz; all data are outputted via a PCI Express endpoint on FPGA1 to a PCI-e bus.

For the PC, a computer with a PCI-e×8 bus and a processor chipset with a DMA (Direct Memory Access) function to transfer data from the PCI-e bus to standard memory such as DRAMs was adopted. Using the DMA function, the input images and processed results from the IDP Express board can be memory-mapped onto the allocated memories on the PC at a rate of 4000 images of 512×512 pixels in a second. Various API functions associated with board control and data access to memory-mapped data are prepared for a 32-bit Windows XP OS as middleware to develop application programs. With these API functions, only the required pixels or processed results can be accessed in real time. In this paper, we used the same PC used in the evaluation in Section IV.

B. Implemented Hardware Logic

We divided a 512×512 image ($N = 512$) into 4096 cells ($M = 64$) of 8×8 pixels ($n = 8$) and implemented our cell-based multi-object feature extraction algorithm as hardware logic on FPGA2 on the IDP Express board. The circuit hardware-implemented on the IDP Express board, consists of a binarization module, a cell-based features calculation module, a cell-based labeling module, and a data selector module for FIFO output. Fig. 10 and 11 show the schematic data flow and the timing chart of the implemented circuit, respectively.

The binarization module can convert 512×512 input images into binary images by scanning in units of 8 pixels from the upper left to the lower right using X and Y address signals with a 75.6 MHz clock. This sub-module can binarize color images with a Bayer filter as well as gray images, and it can convert input images into binary images in parallel for 8 pixels in both cases of color/gray 8-bit images. When a color image sensor is mounted on the camera head, a color conversion sub-module converts RGB images into 8-bit HSV color images [46], hue $H(\boldsymbol{r})$, saturation $S(\boldsymbol{r})$, and value $V(\boldsymbol{r})$, in parallel with 8-pixel data after RGB conversion for input images with a Bayer filter. In this module, HSV images are converted to a binary image $B(\boldsymbol{r})$ by specifying a certain vivid and bright color with four thresholds of θ_{HL}, θ_{HH}, θ_S, and θ_V. When $\theta_{HL} < \theta_{HH}$,

$$B(\boldsymbol{r}) = \begin{cases} 1 & (H \in [\theta_{HL}, \theta_{HH}], S \geq \theta_S, V \geq \theta_V), \\ 0 & (\text{otherwise}); \end{cases} \tag{23}$$

otherwise,

$$B(\boldsymbol{r}) = \begin{cases} 1 & (H \notin (\theta_{HH}, \theta_{HL}), S \geq \theta_S, V \geq \theta_V), \\ 0 & (\text{otherwise}). \end{cases} \tag{24}$$

In the case of a gray image sensor, an input image $I(\boldsymbol{r})$ is converted into a binary image $B(\boldsymbol{r})$ with a threshold θ_L as follows:

$$B(\boldsymbol{r}) = \begin{cases} 1 & (\theta_L \leq I(\boldsymbol{r})), \\ 0 & (\text{otherwise}). \end{cases} \tag{25}$$

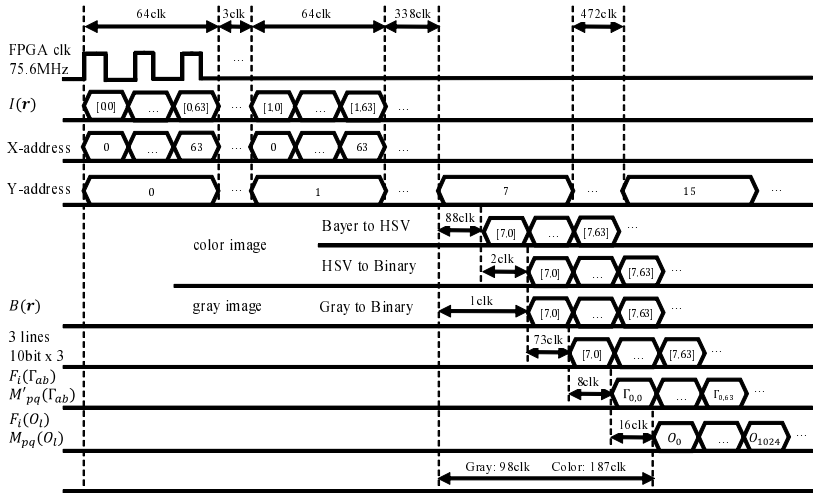


Fig. 11. Timing chart of control signals.

The delay time for obtaining the binary values of 8 pixels is 1 clock (= 13.2 ns) for binary image, and 90 (= 1.19 μs) clocks for color image. All the parameters for binarization can be simultaneously adjusted by updating register values on commands from a PC.

The cell-based features calculation module can calculate 25 HLACs, $F_i(\Gamma_{ab})$ $(i = 0, \ldots, 24)$, and two 1st moments, $M_{10}(\Gamma_{ab})$ and $M_{01}(\Gamma_{ab})$, for 4096 cells Γ_{ab} of 8×8 pixels by using a two-line cache, 27 neighbor-processing sub-modules, and 27 cell-based summation sub-modules. Here, the 0th order moment is equivalent to the 0th order HLAC, and it can be calculated by sharing the circuit for the 0th order HLAC in this module. In the two-line cache, image data on two previous rows of 512 pixels are stored for 3×3 neighbor processing of the HLACs. 64 clocks (= 0.85 μs) are required for scanning a row of 512 pixels, and the delay time in the two-line cache is 73 clocks (= 0.96 μs). The neighbor-processing sub-modules can calculate 25 types of $B(\boldsymbol{r})B(\boldsymbol{r}+\boldsymbol{a}_1)B(\boldsymbol{r}+\boldsymbol{a}_2)$ in Eq. (9), $xB(\boldsymbol{r})$, and $yB(\boldsymbol{r})$, by using three-input AND circuits among the 3×3 neighbor pixels. These values are added as the HLACs and 1st order moments in parallel for 8 pixels by eight-input parallel adders. The cell-based summation sub-module can sequentially add the 8-pixel HLACs or moments for 4096 cells of 8×8 pixels in synchronization with the raster scanning timing of the image. Immediately after all the pixels in a cell are scanned, these sub-modules output 25 cell-based HLACs and two 1st moments with a delay time of 82 clocks (= 1.08 μs) for gray image and 171 clocks (= 2.26 μs) for color image in parallel, which involves the delay time in the two-line cache.

The cell-based labeling module can calculate 25 HLACs, $F_i(O_l)$, and two 1st order moments, $M_{10}(O_l)$ and $M_{01}(O_l)$, for 1024 $(= L)$ labeled regions O_l by using a flag map sub-module, a labeling sub-module, and 27 label-domain summation sub-modules. The flag map sub-module determines active cells among the 4096 cells when the cell-based 0th order moment $M'_{00}(\Gamma_{ab})$ is over than a threshold θ_P; it takes one clock cycle. In the labeling sub-module, all the active cells are labeled such that their label numbers match those of other 4-connected cells. Tentative labels l'_{ab} of 64 cells in a current row are assigned by taking 2 clock cycles. For the relabeling sub-process in Eqs. (18) and (19), tentative labels of 64 cells in the current row are sequentially updated by taking 4 clock cycles to renew memory r and tentatively label l' in the previous row. These sub-processes are executed at every eight rows; its interval is 512 clocks. The label-domain summation sub-module incrementally adds the cell-based HLACs $F_i(\Gamma_{ab})$ and 1st moments $M_{pq}(\Gamma_{ab})$ of active cells to the dedicated memories $f_i(l'_{ab})$ and $m_{pq}(l'_{ab})$ of the same tentative label number l'_{ab} every when the 27 cell-based features for a cell Γ_{ab} is completely calculated. After raster scanning all the pixels in an input image, these sub-modules output 25 label-domain HLACs and two label-domain 1st moments in parallel by taking 16 clock cycles. For each label, 25 19-bit unsigned integers and two 26-bit unsigned integers are outputted as the HLACs and 1st moments, respectively.

TABLE V
RESOURCE CONSUMPTION OF FPGA2

Device Type	Xilinx XC3S5000-4FGG900
Slice	11,684/33,380 (35%)
Slice Flip-Flop	13,125/66,560 (19%)
4-input LUT	12,401/66,560 (18%)
Bounded IOB	228/633 (36%)
Block RAM	91/104 (87%)
MULT18×18	20/104 (19%)
GCLK	3/8 (37%)

In the data selector module, FIFO output for an external PC is selected with X and Y address signals from the input image $I(\boldsymbol{r})$ or the 27 types of label-domain features of 1024 labeled objects. The delay time in calculating the 1024 label-domain HLACs and 1st order moments in the cell-based labeling module is 98 clocks (= 1.29 μs) for gray image and 187 clocks (= 2.47 μs) for color image after raster-scanning all the pixels in an input image, whereas the delay in outputting them to the external PC is one frame (= 0.5 ms).

The circuit module described above, which requires pixel-level computation of 512×512 pixels, was implemented by hardware logic of the FPGA2 on the IDP Express board. The resource consumption of the FPGA was shown in Table V. Fig. 12 shows the pie chart that shows how the block RAM

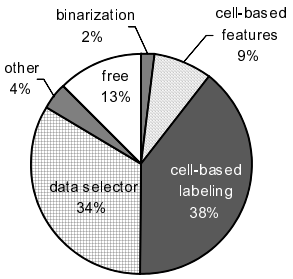


Fig. 12. Consumed block RAM of FPGA2

of the FPGA2 was consumed for the implemented hardware logic. We also confirmed that the 25 HLACs and two 1st moments for 1024 labeled regions can be outputted to the PC for 512×512 images at 2000 fps.

VI. Experiment

A. Tracking of a rotating plate with many letters

To verify the tracking performance of our developed multi-object extraction system, we present experimental results for a rotating planar plate with 21 small letters of 7 mm size, "&" s, and 4 large letters of 17 mm size, "@" s, with all the letters in all cases rotated at 16 rps in the camera view. In this experiment, gray images were binarized with a threshold θ_L = 48. The threshold θ_P to judge active cells was set to 8. The exposure time of the camera head was set to 0.05 ms. Multi-object extraction was executed for 512×512 images at 2000 fps. The 512×512 image corresponded to the area of 85 mm×85 mm size on the rotating plate.

Fig. 13 shows the four input image sequence, the binarized images highlighted by using the label maps, and the sizes and xy positions of labeled objects O_l as circles, taken at intervals of 0.01 s for t = 0.00–0.03 s; the size $A(O_l)$ and the xy position $(c_x(O_l), c_y(O_l))$ of a labeled object O_l were calculated by using their 0th and 1st moments as follows:

$$A(O_l) = M_{00}(O_l), \tag{26}$$

$$(c_x(O_l), c_y(O_l)) = \left(\frac{M_{10}(O_l)}{M_{00}(O_l)}, \frac{M_{01}(O_l)}{M_{00}(O_l)}\right). \tag{27}$$

In the figure, it can be seen that the complex-shaped small and large letters were correctly labeled and that their sizes and centroids corresponded to them.

Fig. 14 shows the numbers of "&"s and "@"s over 0.5 s for t = 0.00–0.50 s. They were counted by using the 0th moments $M_{00}(O_l)$ as follows; "&" when $500 \leq M_{00}(O_l) < 1500$ or "@" when $M_{00}(O_l) \geq 1500$. In the figure, the numbers of letters in the left-half image ($c_x(O_l) < 256$) and those in the right-half image ($c_x(O_l) \geq 256$) are shown, as well as those in the entire image. The numbers of "&"s and "@"s in the half-images were cyclically changed in correspondence with their rotation at 16 rps, while their numbers in an entire image was always constant, with 21 "&"s and 4 "@"s. Fig. 15 shows the sizes and xy positions of the labeled letters of 1–5 in Fig. 13(c) over 0.5 s for t = 0.00–0.50 s. All the labeled objects at the current frame were continuously tracked by searching their xy positions in the small window regions around the xy positions of the labeled objects at the previous frame; the window size

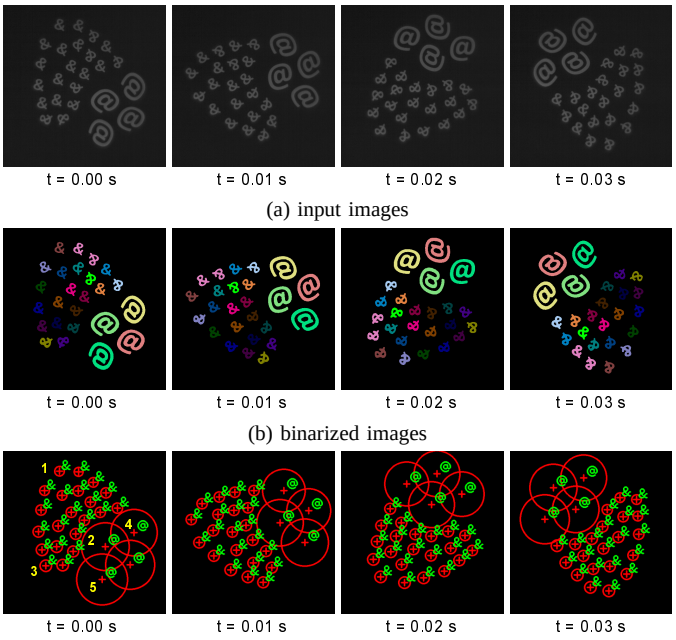


Fig. 13. Input images, binarized images highlighted with label maps, and circles that indicates sizes and xy positions of labeled letters, in tracking a rotating plate with many letters.

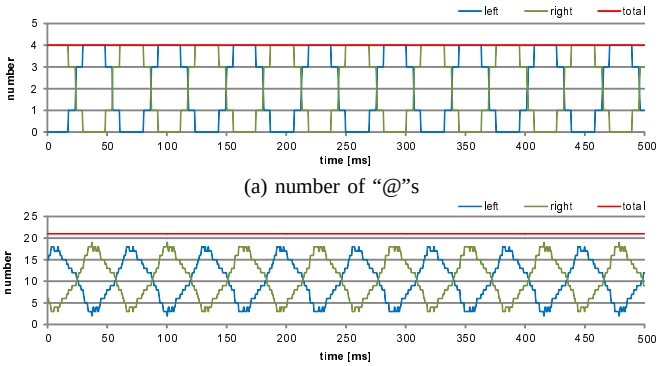
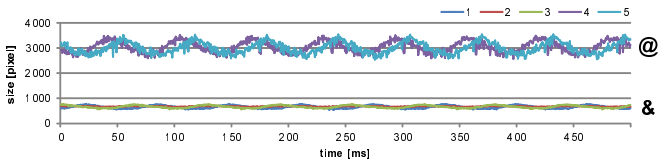


Fig. 14. Number of "@"s and number of "&"s.

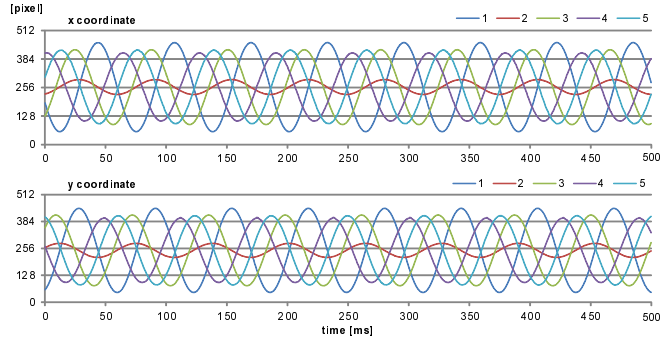
was set to 20×20 pixels. The trajectories of their xy positions were sinusoid curves at 16 Hz and their amplitudes were proportional to the distances between the center of the rotation and the letters, while the trajectories of the sizes of the selected letters were almost constant in time.

B. Recognition of image patterns projected at 1000 fps

To confirm multi-object recognition at a high frame rate by using 25 HLACs, we conducted an experiment on recognition of image patterns projected from a high-speed projector. The frame rate and exposure time of the camera head were set to 1000 fps and 1 ms, respectively. Gray images were binarized with a threshold θ_L = 128. The threshold θ_P to judge active cells was set to 1. Light Commander DLP 5500 (Texas Instruments Inc., Texas, USA) was used as a high-speed projector, and it projected binary images of 1024×768 pixels at 1000 fps in synchronization with our developed multi-object extraction system. 16 binary images were iteratively



(a) sizes



(b) xy positions

Fig. 15. Sizes and xy positions of several labeled letters.

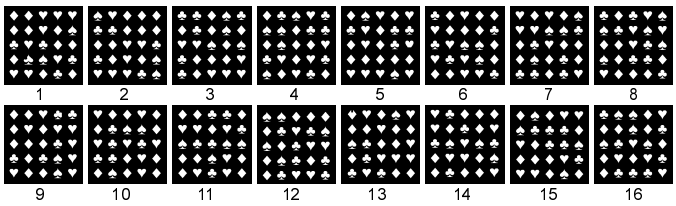
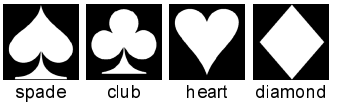


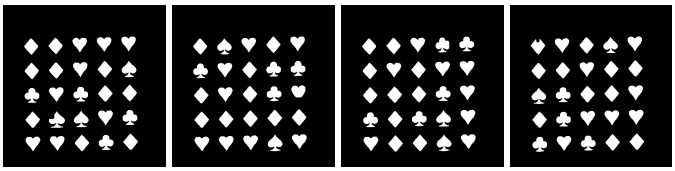
Fig. 16. 16 binary images projected at 1000 fps.



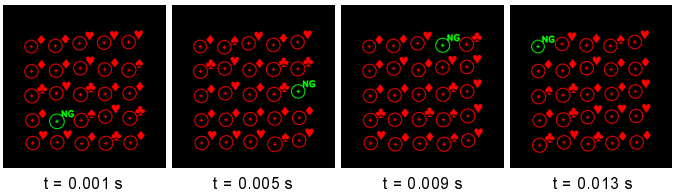
(a) trumpsuits (learned)

(b) NG patterns (unlearned)

Fig. 17. Patterns to be recognized.



(a) binarized images



(b) positions, sizes, and recognized results

Fig. 18. Binarized images, and positions, sizes, and recognized results for labeled objects, in projecting image patterns at 1000 fps.

projected in the order listed in Fig. 16. These images were mapped to the regions of 384×384 pixels in the projection images of 1024×768 pixels. The projected images involve 5×5 objects; they were trumpsuits of spade (pattern 0), club (pattern 1), heart (pattern 2), and diamond (pattern 3) in Fig. 17(a), and several objects had slightly different shapes as no good (NG) patterns in Fig. 17(b). All the four trumpsuits in the projected image had identical areas, and they were difficult to recognize based on the 0th order moment. In the sequence of the projected images, the numbers of trumpsuits changed regularly every 4 images; the number of spade increased from 2 to 5, the number of club increased from 4 to 7, the number of heart decreased from 8 to 5, the number of diamond changed in the order of 10, 10, 9, and 8, and the number of NG pattern changed in the order of 1, 0, 0, and 0.

The 25 HLACs of multiple labeled objects were used for multi-object recognition by the nearest neighbor (NN) method [47]. As a discriminant function for classes i $(=0,\ldots,3)$ of the trumpsuits patterns, we used the distance function $d_i(\boldsymbol{\xi})$ for a 25-dimension HLAC vector $\boldsymbol{\xi}$, normalized by the standard deviation σ_i of the distances between the average vector $\boldsymbol{\mu}_i$ and reference data in class i:

$$d_i(\boldsymbol{x}) = \frac{\|\boldsymbol{x} - \boldsymbol{\mu}_i\|}{\sigma_i}. \tag{28}$$

After calculating the distances for all the classes, we assigned all the labeled objects in an image to a specific class for which the distance was minimum and below a threshold, and the labeled object was assigned to an unknown state if the distances for all the classes exceeded the threshold. Before real-time recognition, 100 sets of the 25 HLACs extracted from the trumpsuits located at different places in an image, were acquired off-line and these patterns were learned for the NN method by using the reference data of 100 sets. The threshold for the discriminant function to determine which class a labeled object belongs to were set to 0.4.

Fig. 18(a) shows the four binarized image sequence, taken at intervals of 4 ms. Fig. 18(b) shows the circles with the recognized results, whose centers and areas correspond to the centroids and sizes of the labeled objects, respectively. This figure shows that the trumpsuit patterns were correctly labeled and recognized at their centroid positions at intervals of 1 ms and that their recognized classes corresponded to their actual classes, including that the NG patterns were recognized as the unknown class. Fig. 19 shows the total number of the labeled objects in an image, the numbers of spades, clubs, hearts, and diamonds, and the number of NG patterns over 0.05 s for t = 0.00–0.05 s. In Fig. 19, the numbers of the recognized trumpsuits changed and an NG pattern was detected at intervals of 4 ms by using 25 HLACs, while the total number of the labeled objects was 25 constantly. This trend corresponded to the 1000-fps image projection, in which the numbers of the trumpsuits changed regularly every 4 images.

C. Recognition of quick hand gestures in RPS game

We conducted an experiment on real-time gesture recognition by using 25 HLACs: shapes of two human hands were recognized in rock-paper-scissors (RPS) game. The frame rate and exposure time of the camera head were set to 1000 fps and

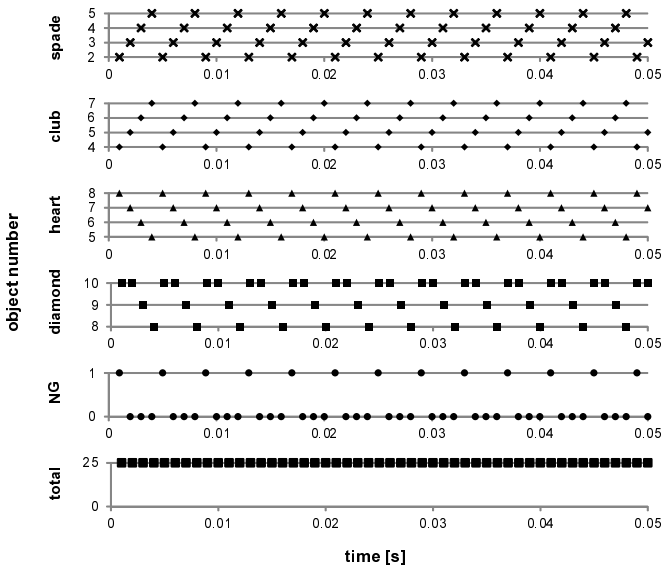


Fig. 19. Total number of labeled objects, numbers of spades, clubs, hearts, and diamonds, and number of NG patterns.

1 ms, respectively. Color images were binarized by extracting skin color with thresholds $\theta_{HL} = 0°$, $\theta_{HH} = 43°$, $\theta_S = 1$, and $\theta_V = 42$. Here the observable ranges of hue, saturation, and value were set to [0°, 360°], [0,255], and [0,255], respectively. The threshold θ_P to judge active pixels was set to 1, and we extracted the labeled objects for recognition only when their 0th moments were over than 1000. In the experiment, the labeled objects were recognized as "rock" (class 0), "paper" (class 1), and "scissors" (class 2) in RPS game as shown in Fig. 20. Two human hands in an image were classified by using the NN method with the discriminant function in Eq. (28) in the same way as the experiment in Subsection VI-B. Before real-time recognition, 60 sets of the 25 HLACs extracted from the hand shapes at different places in an image, were acquired off-line. They were used as reference data to determine the discriminant function in Eq. (28). The threshold for the discriminant function was set to 2.8 to classify a labeled object in a specific class, including an unknown class.

Fig. 21 shows the seven input image sequence, the binarized images, and the circles with the recognition results, whose centers and areas correspond to the centroids and sizes of the labeled objects, respectively, taken at intervals of 0.4 s. In the experiment, two persons played the RPS game with three battle times over 3 s. It can be observed that the hand shapes of two persons were correctly classified. Fig. 22 shows that the xy positions of the two labeled objects over 3 s for $t = 0.0$–3.0 s; they corresponded to the hands of the two persons, and it can be seen that the up-and-downs in the y positions were synchronized with each other for the battles in RPS game. Fig. 23 shows the temporal changes of the three discriminant functions of "rock", "paper", and "scissors" over 3 s for the labeled two hand regions. Fig. 24 shows the classified results of the two labeled hands and victory or defeat results in the RPS game over 3 s. Depending on the smallest distance between the labeled region and the classes, the hand shapes corresponded their classes of "rock", "paper", and "scissors", correctly especially when there were battles around $t = 0.75$ s, 1.65 s, and 2.45 s. These battle timings were

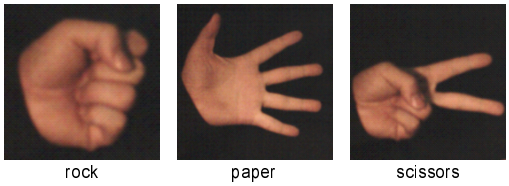
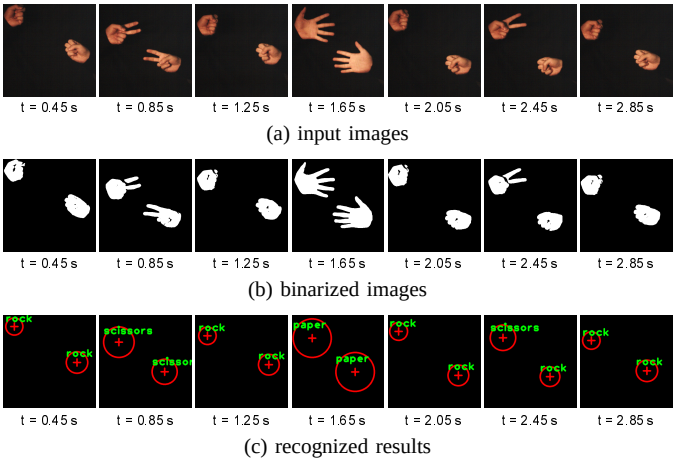


Fig. 20. "rock", "paper", and "scissors" in RPS game.



(a) input images

(b) binarized images

(c) recognized results

Fig. 21. Input images, binarized images, and recognized results in RPS game.

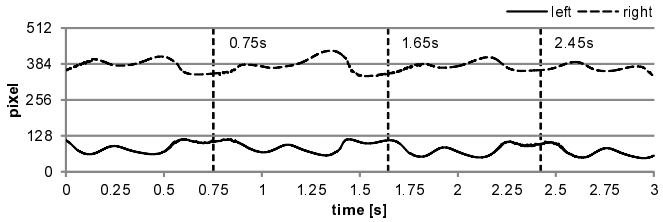
determined manually. Thus, it can be observed that our multi-object extraction system can simultaneously know who wins the RPS game by recognizing the complicated hand shapes based on the 25 HLACs at a high frame rate.

VII. Conclusion

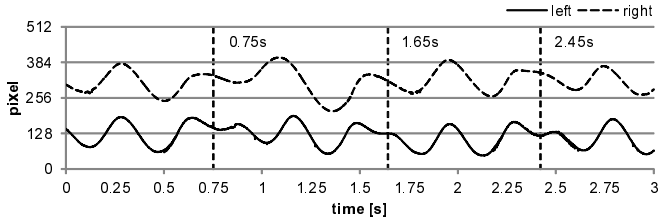
In this study, we developed an HFR multi-object extraction system by hardware-implementing a cell-based multi-object feature extraction algorithm; it can extract sizes, positions, and HLACs of 1024 labeled objects in an image by dividing a 512×512 image into 64×64 cells, and it could perform multi-object extraction of 512×512 images in real time at 2000 fps. Its performance was verified in several experiments; tracking for multiple objects rotating at 16 rps, multiple pattern recognition for image patterns projected from a high-speed projector, and human gesture recognition in RPS game. Our cell-based multi-object feature extraction algorithm introduced in this study can control memory consumption and processing speed by adjusting cell size and maximum label number. This property enables to implement this algorithm on cheap FPGA devices for higher resolution images in embedded systems easily. Thus, we plan to extend such HFR multi-object tracking and recognition to various embedded applications such as motion capture and blob analysis in factory automation and robotics.

References

[1] A. Rosenfeld and J. L. Pfaltz, "Sequential operation in digital picture processing," *J. Assoc. Comput.*, pp. 471–494, 1966.
[2] D. Wang, "Unsupervised video segmentation based on watersheds and temporal tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 5, pp. 539–546, 2000.
[3] G. Hamarneh and X. Li, "Watershed segmentation using prior shape and appearance knowledge," *Image Vis. Comput.*, vol. 27, pp.59–68, 2009.

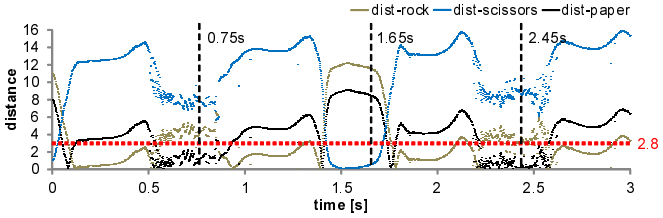


(a) x position

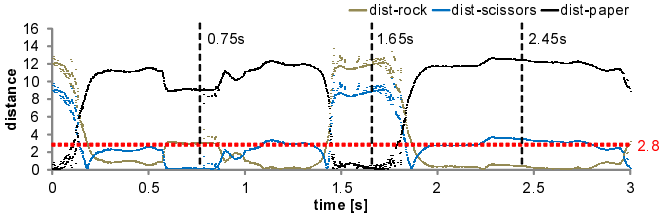


(b) y position

Fig. 22. xy positions of two regions labeled as human hands.

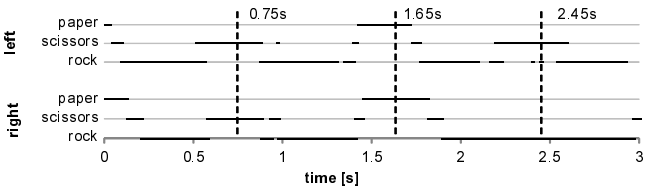


(a) discriminant functions for a hand in the left

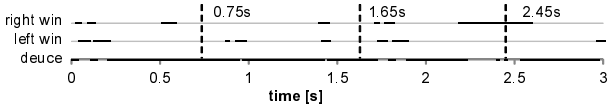


(b) discriminant functions for a hand in the right

Fig. 23. Discriminant functions in RPS game.



(a) classified results



(b) victory or defeat results

Fig. 24. Recognized results in RPS game.

[4] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, "Watershed cuts: Minimum spanning forests and the drop of water principle," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 8, pp. 1362–1374, 2009.

[5] Y. Boykov and M-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D Images," *Proc. Int. Comput. Vis.,*, pp. I:105–112, 2001.

[6] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut – Interactive foreground extraction using iterated graph cuts," presented at *the ACM SIGGRAPH*, 2004.

[7] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient N-D image segmentation," *Int. J. Comput. Vis.*, vol. 70, no. 2, pp. 109–131, 2006.

[8] N. Lerme, F. Malgouyres, and L. Letocart, "Reducing graphs in graph cut segmentation," *Proc. IEEE Int. Conf. Image Process.*, pp. 3045–3048, 2010.

[9] I. Kompatsiaris and M.G. Strintzis, "Spatiotemporal segmentation and tracking of objects for visualization of videoconference image sequences," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 8, pp. 1388–1403, 2000.

[10] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, 2002.

[11] J. Fan, M. Han, and J. Wang, "Single point iterative weighted fuzzy c-means clustering algorithm for remote sensing image segmentation," *Pattern Recognition*, vol. 42, no. 11, pp. 2527–2540, 2009.

[12] G. Kuntimad and H.S. Ranganath, "Perfect segmentation using pulse coupled neural networks,h *IEEE Trans. Neural Netw.*, vol. 10, no. 3, pp. 591-598, 1999.

[13] K. Yao, M. Mignotte, C. Collet, P. Galerne, and G. Burel, "Unsupervised segmentation using a self-organizing map and a noise model estimation in sonar imagery," *Patttern Recognition Lett.*, vol. 33, no. 9, pp. 1575-1584, 2000.

[14] Z. Xiao, J. Shi, and Q. Chang, "Automatic image segmentation algorithm based on PCNN and fuzzy mutual information," *Proc. IEEE Int. Conf. Comput. Info. Tech.*, pp. 241–245, 2009.

[15] Y. Chen, S.K. Park, Y.D. Ma, and R. Ala, "A new automatic parameter setting method of a simplified PCNN for image segmentation," *IEEE Trans. Neural Netw.*, vol. 22, no. 6, pp. 880–892, 2011.

[16] T. M. Bernard, B. Y. Zavidovique, and F. J. Devos, "A programmable artificial retina," *IEEE J. Solid-State Circuits*, vol. 28, no. 7, pp. 789–797, 1993.

[17] J. E. Eklund, C. Svensson, and A. Astrom, "VLSI implementation of a focal plane image processor - A realization of the near-sensor image processing concept," *IEEE Trans. VLSI Systems*, vol. 4, no. 3, pp. 322–335, 1996.

[18] T. Komuro, S. Kagami, and M. Ishikawa, "A dynamically reconfigurable SIMD processor for a vision chip," *IEEE J. Solid-State Circ.*, vol. 39, no. 1, pp. 265–268, 2004.

[19] I. Ishii, K. Yamamoto, and M. Kubozono, "Higher order autocorrelation vision chip," *IEEE Trans. Electron Dev.*, vol. 53, no. 8, pp. 1797–1804, 2006.

[20] Y. Watanabe, T. Komuro, and M. Ishikawa, "955-fps real-time shape measurement of a moving/deforming object using high-speed vision for numerous-point analysis," *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 3192–3197, 2007.

[21] S. Hirai, M. Zakoji, A. Masubuchi, and T. Tsuboi, "Realtime FPGA-based vision system," *J. Robot. Mechat.*, vol. 17, no. 4, pp. 401–409, 2005.

[22] I. Ishii, R. Sukenobe, T. Taniguchi, and K. Yamamoto, "Development of high-speed and real-time vision platform, H^3 Vision," *Proc. IEEE/RSJ Int. Conf. Intelli. Rob. Sys.*, pp. 3671–3678, 2009.

[23] Q. Gu, T. Takaki, and I. Ishii, "A fast multi-object extraction algorithm based on cell-based connected components labeling," *IEICE Trans. Inf. Syst.*, vol. E95-D, no. 2, pp. 636–645, 2012.

[24] N. Otsu et al., "A new scheme for practical, flexible, and intelligent vision systems," *Proc. IAPR Workshop Comput. Vis.*, pp. 431–435, 1988.

[25] R. M. Haralick, "Some neighborhood operations, in: Real Time/Parallel Computing," *Image Analysis, M. Onoe, K. Pres-ton, A. Rosenfled, (Eds.), Plenum Press, New York.*, pp. 11–35, 1981.

[26] K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labeling based on sequential local operations," *Comput. Vis. Image Underst.*, vol. 89, pp. 1–23, 2003.

[27] L. He, Y. Chao, and K. Suzuki, "A run-based two-scan labeling algorithm," *Comput. Vis. Image Underst.*, vol. 17, no. 5, pp. 749–756, 2008.

[28] L. He, Y. Chao, K. Suzuki, and K. Wu, "Fast connected-component labeling," *Pattern Recognition*, vol. 42, pp. 1977–1987, 2008.

[29] E. Mandler and M. F. Oberlander, "One-pass encoding of connected components in multi-valued images," *Proc. IEEE Conf. Comput. Vis. Pattern Recognition*, pp. 64–69, 1990.

[30] F. Chang, C.-J. Chen, and C.-J. Lu, "A linear-time component labeling algorithm using contour tracking technique," *Comput. Vis. Image Underst.*, vol. 93, no. 2, pp. 206–220, 2004.

[31] C. J. Nicol, "A systolic approach for real time connected component labeling," *Comput. Vis. Image Underst.*, vol. 61, pp. 17–31, 1995.

[32] A. Baumker and W. Dittrich, "A new parallel MIMD connected com-

ponent labeling algorithm," *Proc. Int. Parallel Process. Symp.*, pp. 429–433, 1996.
[33] V. Chaudhary and J. K. Aggarwal, "Parallel image component labeling for target acquisition," *Opt. Eng.*, vol. 37, no. 7, pp. 2078–2090, 1998.
[34] K. Benkrid, S. Sukhsawas, D. Crookes, and A. Benkrid, "An FPGA-Based Image Connected Component Labeller," *Lecture Notes in Computer Science.*, pp. 1012-1015, 2003.
[35] H. Flatt, S. Blume, S. Hesselbarth, T. Schunemann, and P. Pirsch, "A parallel hardware architecture for connected component labeling based on fast label merging," *Proc. IEEE Int. Conf. Application-Specific Systems, Architectures and Processors.*, pp. 144–149, 2008.
[36] K. Appiah, A. Hunter, P. Dickinson, and J. Owens, "A run-length based connected component algorithm for FPGA implementation," *Proc. IEEE Int. Conf. Field-Programmable Technology.*, pp. 177–184, 2008.
[37] K. Appiah, A. Hunter, P. Dickinson, and H. Meng, "Accelerated hardware video object segmentation: From foreground detection to connected components labelling," *Comput. Vis. Image Underst.*, vol. 114, pp. 1282–1291, 2010.
[38] D.G. Bailey and C.T. Johnston, "Single pass connected components analysis," *Proc. Image Vis. Comput. New Zealand*, , pp. 282–287, 2007.
[39] C.T. Johnston and D.G. Bailey, "FPGA implementation of a single pass connected components algorithm," *Proc. IEEE Int. Symp. Electronics, Design, Test and Applications*, pp. 228–231, 2008.
[40] V.S. Kumar, K. Irick, A. Al Maashri, and N. Vijaykrishnan, "A scalable bandwidth aware architecture for connected component labeling," *Proc. IEEE Annu. Symp. VLSI*, pp. 116-121, 2010.
[41] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man Cybern.*, vol. 9, no. 1, pp. 62–66, 1979.
[42] I. Ishii, T. Tatebe, Q. Gu, Y. Moriue, T. Takaki and K. Tajima, "2000 fps real-time vision system with high-frame-rate video recording," *IEEE Int. Conf. Rob. Autom.*, pp. 1536–1541, 2010.
[43] Q. Gu, T. Takaki, and I. Ishii, "2000-fps multi-object extraction based on cell-based labeling," *Proc. IEEE Int. Conf. Image Process.*, pp. 3761–3764, 2010.
[44] I. Ishii, R. Sukenobe, K. Yamamoto, and T. Takaki, "Real-time image recognition using HLAC features at 1000 fps," *Proc. IEEE Int. Conf. Rob. Biomimetics*, pp. 954–959, 2009.
[45] F.S. Samaria and A.C. Harter, "Parameterisation of a stochastic model for human face identification," *Proc. IEEE Workshop Appli. Comput. Vis.*, pp. 138–142, 1994.
[46] A. R. Smith, "Color Gamut Tranform Pairs," *SIGGRAPH Comput. Graphics*, vol. 12, no. 3, pp. 12–19, 1978.
[47] T.M. Cover and P.E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inform. Theory*, vol. 13, no. 1, pp. 21–27, 1967.