IEICE TRANS. INF. & SYST., VOL.E95–D, NO.2 FEBRUARY 2012

PAPER

A Fast Multi-Object Extraction Algorithm Based on Cell-Based Connected Components Labeling

Qingyi GU[†a)], **Takeshi TAKAKI**[†b)], *Nonmembers*, *and* **Idaku ISHII**[†c)], *Member*

SUMMARY We describe a cell-based connected component labeling algorithm to calculate the 0th and 1st moment features as the attributes for labeled regions. These can be used to indicate their sizes and positions for multi-object extraction. Based on the additivity in moment features, the cell-based labeling algorithm can label divided cells of a certain size in an image by scanning the image only once to obtain the moment features of the labeled regions with remarkably reduced computational complexity and memory consumption for labeling. Our algorithm is a simple-one-time-scan cell-based labeling algorithm, which is suitable for hardware and parallel implementation. We also compared it with conventional labeling algorithms. The experimental results showed that our algorithm is faster than conventional raster-scan labeling algorithms.

key words: *fast multi-object extraction, connected component labeling*

1. Introduction

Connected component labeling is a basic method that is used to extract multiple objects from an image as segmented regions. Many labeling algorithms have been used to calculate their image features such as number of labeled regions, size, and position for practical machine inspection in factory automation, biomedical, and other applications. In most cases, they have been applied to video images with video signal formats (e.g., NTSC 30 fps, PAL 25 fps); however, there is a strong requirement for simultaneous labeling and feature extraction to capture and recognize rapidly moving objects in an image that are too fast for the human eye to see.

Many types of high-speed vision systems that can operate at high frame rates of 1000 fps or more have been developed for simultaneous video processing. Vision chips [1]–[4] are on-chip solutions for simultaneous video processing; these are produced by fabricating 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. For example, massively parallel co-processors have been implemented for multi-target tracking [5]; Hough transforms have been implemented on an FPGA [6]; and a high-speed vision platform for 1024×1024 pixel images [7] has been developed to execute real-time image processing algorithms such as color marker tracking, feature point tracking, and optical flow estimation. If we could implement a real-time image processing function to extract image features of labeled regions in an image on such high-speed vision platforms, it would become possible to carry out multi-object extraction at a higher frame rate than the NTSC frame rate of 30 fps.

However, conventional labeling algorithms are not always suitable for fast multi-object extraction because most of them require a large memory area, depending on the image size with sequential scanning for an entire image; this makes it difficult to carry out high-speed image processing on a high-speed vision platform. In this study, a cell-based connected component labeling algorithm that can remarkably reduce computational complexity and memory consumption is proposed to accelerate moment features calculation of the labeled regions in an image. Section 2 summarizes previous works for connected component labeling and their problems in high-speed implementation. Section 3 introduces a cell-based labeling algorithm that can segment all the divided regions of a certain size in an image to obtain their 0th and 1st moment features; these moment features can express their sizes and positions for multi-object extraction. In Sect. 4, the execution times and accuracies of our cell-based labeling algorithm on a personal computer (PC) are evaluated for several image patterns, and these results are compared with those obtained for several conventional connected component labeling algorithms. Memory consumption of our algorithm is also discussed in Sect. 4.

2. Previous Works

Rosenfeld's algorithm [8] is one of the earliest works on connected component labeling with sequential image processing. This algorithm carries out two passes over an entire image to relabel and temporarily store label equivalences as an image until the second scanning is completed. To reduce the number of scanned pixels for labeling, Haralick proposed an improved algorithm [9] that does not store the label equivalences explicitly by iterating the forward-and-backward scans alternatively until no further merging is possible. This algorithm cannot always guarantee its execution time in the worst case for real-time applications because the number of image scans varies with the complexity of the connected components. To solve this problem in connected component labeling with bidirectional scanning, several improved algorithms [10], [11] that introduce relabeling tables to Haralick's algorithm have recently been proposed.

Manuscript received April 11, 2011.
Manuscript revised September 8, 2011.

[†]The authors are with the Department of System Cybernetics, Hiroshima University, Higashihiroshima-shi, 739–8527 Japan.
a) E-mail: gu@robotics.hiroshima-u.ac.jp
b) E-mail: takaki@robotics.hiroshima-u.ac.jp
c) E-mail: iishii@robotics.hiroshima-u.ac.jp

DOI: 10.1587/transinf.E95.D.636

To reduce the time for assigning new labels, He et al. have proposed a fast two-scan run-based algorithm [12] that considers the connectivity between runs instead of pixels. Although this algorithm can be accelerated by reducing the number of relabeling operations using equivalent-label sets and a representative label table, it is only efficient for run-length-encoded images and its computational complexity increases when general images are handled. An improved fast two-scan algorithm [13] has been also proposed by simplifying the label assignment strategy to avoid the calculation for small connected components. However, it still requires high memory consumption and two-scan at least, which are disadvantages in accelerating labeling process.

One-scan labeling algorithms [14], [15] have been proposed that would scan connected components in an image by tracking their contours. Their computational speeds depend on the complexity of the connected components and they require that an image should be initially stored to track the contours of connected components with complex trajectories. However, these algorithms are not suitable for hardware implementation because of their irregular image memory access. A one-scan labeling algorithm [16] for hardware implementation has been also reported, while it theoretically requires 50% horizontal blanking period to solve label equivalences after scanning one line. Many parallelized labeling algorithms have been proposed for implementation on various types of parallel computers [17]–[19]; these algorithms can label sub-images on divided block regions of an image in parallel and merge the labeled results into a labeled image for the entire area. 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 temporarily during processing.

Most of the above algorithms have been designed to obtain a labeled image by scanning all the pixels in an image; however, there are many cases in which only the image features of labeled connected components are required, such as in blob processing. Amir et al. developed a hardware implementation of a labeling algorithm [20] with a single image scan that processes each scan line as it becomes available to obtain only the 0th and 1st moment features and bounding boxes of connected components in an image for the pupil tracking of human eyes. They also developed an embedded system for human eye detection by hardware-implementing their algorithm; this system can process a 640×480 pixel image in real-time at 60 fps. Gabbur et al. [21] extended this algorithm by introducing tree structures to merge equivalent labels. Although these algorithms do not need to temporarily store input images and their labeled images during labeling, there exist many constraints that limit their application to multi-object extraction at a high frame rate of 1000 fps or more; (1) sequential image scan for all pixels in an image, which make difficulties to parallelize labeling process at high speed; (2) too many label equivalences for connected components when there exist non-convex objects and isolated points in an image, which often require large memories of image size or more to store image features of the labeled regions.

3. Cell-based labeling algorithm

3.1 Concept

By cropping the image to be scanned with a certain interval, we can easily speed up image feature extraction for connected components in an image, while the extracted image features are degraded in space resolution. Therefore, in this study, we introduce a concept of cell-based labeling to extract the 0th and 1st moment features of connected components; 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.

Our designed algorithm based on the concept of cell-based labeling can obtain moment features $M_{pq}(O_l)$ for labeled regions O_l $(l = 0, \ldots, L-1)$ in a binary image $B(x, y)$ of $N \times N$ pixels; $B(x, y)$ is divided into M^2 cells Γ_{ab} $(a = 0, \ldots, M-1, b = 0, \ldots, M-1)$ of $n \times n$ pixels when $N = nM$,

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

Two processes are included in our algorithm, as shown in Fig. 1: (1) calculation of moment features $M_{pq}(\Gamma_{ab})$ for M^2 cells, which requires pixel-level computation having a complexity of the order $O(N^2)$, and (2) labeling M^2 cells with updated moment features $M_{pq}(O_l)$ for L labeled regions O_l, which requires cell-level computation having a complexity of the order $O(M^2)$. In the case of 4-neighbor connected components, our algorithm only requires $M^2/2$ label equivalences at maximum in labeling process, whereas pixel-based labeling algorithms require $N^2/2$ label equivalences at maximum.

Therefore, by exchanging its computation sequence in labeling process for cells, the computational complexity and label equivalences required for labeling can be remarkably reduced in cell-based computation of the order $O(M^2)$. Here moment features of connected components extracted by our cell-based labeling algorithm are not always matched with those of pixel-based connected component labeling algorithms when their cell size is larger than 1×1 pixel, that is, $n > 1$, because the pixels belonging to different connected components in the same cell or neighboring cells may be often misidentified as the same labeled regions based on the

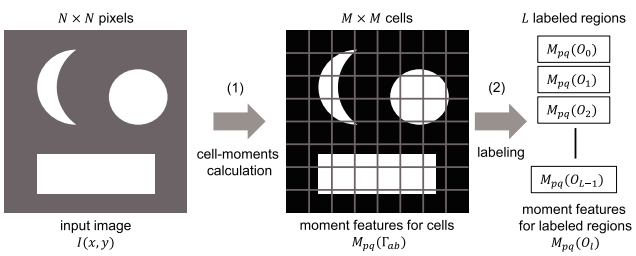


Fig. 1 Concept of cell-based labeling algorithm.

connectivity of cells. This means that our cell-based labeling algorithm is not perfectly equivalent to pixel-based connected component labeling algorithms, and we have to consider a trade-off relationship between its computational complexity and equivalency to pixel-based connected component labeling.

3.2 Additivity in Moment Feature Calculation

The p-th and q-th moment features ($(p,q) = (0,0)$, $(1,0)$, $(0,1)$) for x and y in a region Γ for an image $B(x,y)$ are written as,

$$M_{pq}(\Gamma) = \sum_{(x,y)\in\Gamma} x^p y^q B(x,y). \tag{2}$$

When a region Γ can be divided into K sub-regions $\Gamma_0, \ldots, \Gamma_{K-1}$, moment features $M_{pq}(\Gamma)$ can be calculated by adding the moment features $M_{pq}(\Gamma_k)$ for sub-regions $\Gamma_0, \ldots, \Gamma_{K-1}$ as follows,

$$M_{pq}(\Gamma) = \sum_{k=0}^{K-1} M_{pq}(\Gamma_k). \tag{3}$$

This additivity in moment feature calculation implies that the moment feature for an image region can be obtained by calculating the moment features for its divided sub-regions and then adding them, which is the process of commutativity in moment feature calculation. Based on this commutativity, our algorithm can reduce its computational complexity by exchanging its computation sequence for the connected component labeling of divided cells after calculating the moment features for $n \times n$ pixel cells.

3.3 Algorithm

3.3.1 Calculation of Cell-Based Moment Features

First, the cell-based moment features $M'_{pq}(\Gamma_{ab})$ are calculated for every cell Γ_{ab} of $n \times n$ pixels,

$$M'_{pq}(\Gamma_{ab}) = \sum_{s=0}^{n-1}\sum_{t=0}^{n-1} s^p t^q B(an+s, bn+t). \tag{4}$$

Here, we use a feature $M'_{pq}(\Gamma_{ab})$ 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,

$$M_{00}(\Gamma_{ab}) = M'_{00}(\Gamma_{ab}), \tag{5}$$

$$M_{10}(\Gamma_{ab}) = M'_{10}(\Gamma_{ab}) + an \cdot M'_{00}(\Gamma_{ab}), \tag{6}$$

$$M_{01}(\Gamma_{ab}) = M'_{01}(\Gamma_{ab}) + bn \cdot M'_{00}(\Gamma_{ab}). \tag{7}$$

3.3.2 Labeling of Cells with Updated Moment Features

Next, all the cells are labeled by using the calculated cell-based moment features in order to extract the moment features of the labeled regions as connected components in an image. The labeling process for the cells is carried out by scanning P_{ab} of $M \times M$ cells from the upper left cell to the lower right one. Here P_{ab} is a flag map of $M \times M$ cells to judge whether a cell Γ_{ab} should be labeled or not. In this study, this flag map is defined by checking $M'_{00}(\Gamma_{ab})$ with a threshold θ, which is the sum of the active pixels in Γ_{ab},

$$P_{ab} = \begin{cases} 1 & (M'_{00}(\Gamma_{ab}) \geq \theta) \\ 0 & (\text{otherwise}) \end{cases}. \tag{8}$$

By adjusting a threshold θ, robustness and sensitivity in connected component labeling can be controlled in our algorithm. When θ is large, we can eliminate isolated noises and undesired effects due to other noises in labeling process with little accuracy degradation in moment feature calculation because some non-active cells are ignored although there exist active pixels in the cells. When θ is small, moment features are almost accurately calculated for connected components whereas small connected components caused by noises may adversely affect in labeling process, as shown in pixel-based connected component labeling algorithms.

When Γ_{ab} is a current cell in scanning, 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 $O_1, \ldots, O_L$, $m_{00}(l)$, $m_{10}(l)$, and $m_{01}(l)$ $(l = 0, \ldots, L-1)$ are used as memories to update the 0th and 1st moment features 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 in our algorithm includes (1) a tentative labeling sub-process with updated moment features in every cell, and (2) a relabeling sub-process in every row. In this study, our algorithm is designed to label cells based on 4-neighbor connectivity. Moreover, this cell-based labeling algorithm can be easily expanded to 8-neighbor connectivity by using upper-left, upper, upper-right, and left cells to assign a new label to the current cell, whereas upper and left cells are used in label assignment based on 4-neighbor connectivity.

(1) Tentative labeling sub-process in every cell

When a flag $P_{ab} = 1$, 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 moment features $m_{00}(l'_{ab})$, $m_{10}(l'_{ab})$, and $m_{01}(l'_{ab})$, we consider the following five cases shown in Fig. 2.

Case 1) $P_{ab-1} = P_{a-1b} = 0$.

When both the upper and the left cells are not active, a

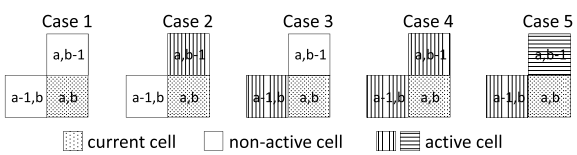


Fig. 2 Relationship between neighboring cells in tentative labeling.

current cell Γ_{ab} obtains a new label l_{new} as a new connected component, which is not assigned to any other cell, and $m_{pq}(l'_{ab})$ for a new labeled region $O_{l_{new}}$ are set to the cell-based moment features $M_{pq}(\Gamma_{ab})$ of the current cell Γ_{ab}.

(A) determine tentative label

$$l'_{ab} = l_{new}. \tag{9}$$

(B) update moment features

$$m_{pq}(l'_{ab}) = M_{pq}(\Gamma_{ab}). \tag{10}$$

Case 2) $P_{ab-1} = 1, P_{a-1b} = 0$.

When only the upper cell Γ_{ab-1} is active, a current cell Γ_{ab} succeeds the label l_{ab-1} of its upper cell. Based on the additivity in moment feature calculation, $m_{pq}(l'_{ab})$ are updated by adding $\hat{m}_{pq}(l'_{ab})$ to $M_{pq}(\Gamma_{ab})$ of the current cell. Here $\hat{m}_{pq}(l)$ is defined as $m_{pq}(l)$ in the previous cell in scanning.

(A) determine tentative label

$$l'_{ab} = l_{ab-1}. \tag{11}$$

(B) update moment features

$$m_{pq}(l'_{ab}) = \hat{m}_{pq}(l'_{ab}) + M_{pq}(\Gamma_{ab}). \tag{12}$$

Case 3) $P_{ab-1} = 0, P_{a-1b} = 1$.

When only the left cell Γ_{a-1b} is active, a current cell succeeds the tentative label l'_{a-1b} of its left cell and $m_{pq}(l'_{ab})$ are updated by adding $\hat{m}_{pq}(l'_{ab})$ to $M_{pq}(\Gamma_{ab})$ in the same manner as Case 2.

(A) determine tentative label

$$l'_{ab} = l'_{a-1b}. \tag{13}$$

(B) update moment features

$$m_{pq}(l'_{ab}) = \hat{m}_{pq}(l'_{ab}) + M_{pq}(\Gamma_{ab}). \tag{14}$$

Case 4) $P_{ab-1} = P_{a-1b} = 1$, and $l_{ab-1} = l'_{a-1b}$.

When both the upper and the left cells are active and they have a common label, a current cell succeeds the tentative label l'_{a-1b} of its left cell as the common label. $m_{pq}(l'_{ab})$ are updated by adding $\hat{m}_{pq}(l'_{ab})$ to $M_{pq}(\Gamma_{ab})$ in the same manner as Case 2.

(A) determine tentative label

$$l'_{ab} = l'_{a-1b}. \tag{15}$$

(B) update moment features

$$m_{pq}(l'_{ab}) = \hat{m}_{pq}(l'_{ab}) + M_{pq}(\Gamma_{ab}). \tag{16}$$

Case 5) $P_{ab-1} = P_{a-1b} = 1$, and $l_{ab-1} \neq l'_{a-1b}$.

When both the upper and the left cells are active and they have different labels, a current cell Γ_{ab} selects a smaller label in the neighboring cells. Then, $m_{pq}(l'_{ab})$ are updated by adding both $\hat{m}_{pq}(l'_{a-1b})$ and $\hat{m}_{pq}(l_{ab-1})$ for the previous two labels to $M_{pq}(\Gamma_{ab})$ based on the additivity in moment feature calculation. Additionally, a memory $r(l)$ is updated for the relabeling process to record which labels are unified.

(A) determine tentative label

$$l'_{ab} = \min(l_{ab-1}, l'_{a-1b}). \tag{17}$$

(B) update moment features

$$m_{pq}(l'_{ab}) = \hat{m}_{pq}(l_{ab-1}) + \hat{m}_{pq}(l'_{a-1b}) + M_{pq}(\Gamma_{ab}). \tag{18}$$

(C) update memory $r(l)$

$$r(\max(l_{ab-1}, l'_{a-1b})) = l'_{ab}. \tag{19}$$

In sub-process (1), $\hat{m}_{pq}(l)$ 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 non-active cells in this study.

(2) Relabeling sub-process in every row

After all the cells in the current row are scanned for sub-process (1), their tentative labels l'_{ab} still remain to be updated while the moment features $m_{pq}(l'_{ab})$ for labeled regions have already been updated. Here the labels for all the cells in the current row should be memorized as upper cell data before starting to label cells in next row. In Case 5, the connected cells in the same row have different labels in the tentative labeling of sub-process (1) when "U" shape component is handled as shown in Fig. 3. In this study, we solved this problem by updating tentative labels l'_{ib} 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,

$$\begin{matrix} r(l'_{ib}) = r(r(l'_{ib})) \\ l_{ib} = r(l'_{ib}) \end{matrix} \quad (i = 0, \ldots, M-1). \tag{20}$$

Here the number of unified tentative labels may increase in sub-process 2-a) when Case 5 often occurs. Therefore, memory areas of moment features for abandoned tentative labels also increase, and they may disturb high-speed

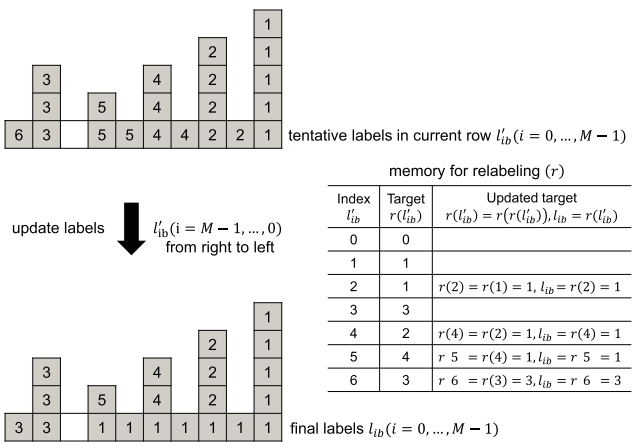


Fig. 3 Relabeling process.

processing and hardware-implementation of cell-based labeling in the worst case. To reduce the memory areas for the abandoned labels, they are released for recycling as new labels after the relabeling process. The memory $r(l)$ for relabeling is then reset as follows,

$$r(l) = l \quad (l = 0, \cdots, L-1). \tag{21}$$

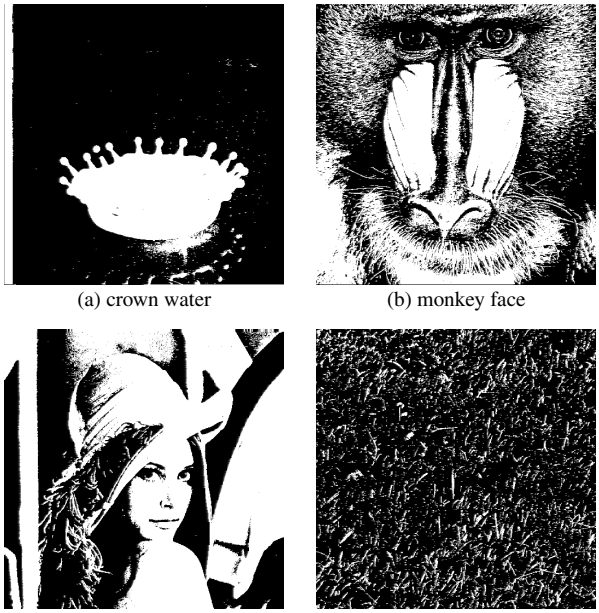
After scanning all the rows for sub-processes (1) and (2), moment features $M_{pq}(O_l)$ for labeled regions O_l are finally obtained as follows,

$$M_{pq}(O_l) = m_{pq}(l). \tag{22}$$

By using the 0th and 1st moment features $M_{pq}(O_l)$ for labeled regions O_l, we can calculate the centroids (x_l, y_l) and areas S_l of all the labeled regions as the positions and sizes for multi-object extraction in an image as follows,

$$(x_l, y_l) = \left(\frac{M_{10}(O_l)}{M_{00}(O_l)}, \frac{M_{01}(O_l)}{M_{00}(O_l)}\right), \tag{23}$$

$$S_l = M_{00}(O_l). \tag{24}$$



(a) crown water (b) monkey face

(c) Lena (d) grass texture

Fig. 4 image patterns in evaluating execution times on a PC.

4. Evaluations

4.1 Execution Time on a PC

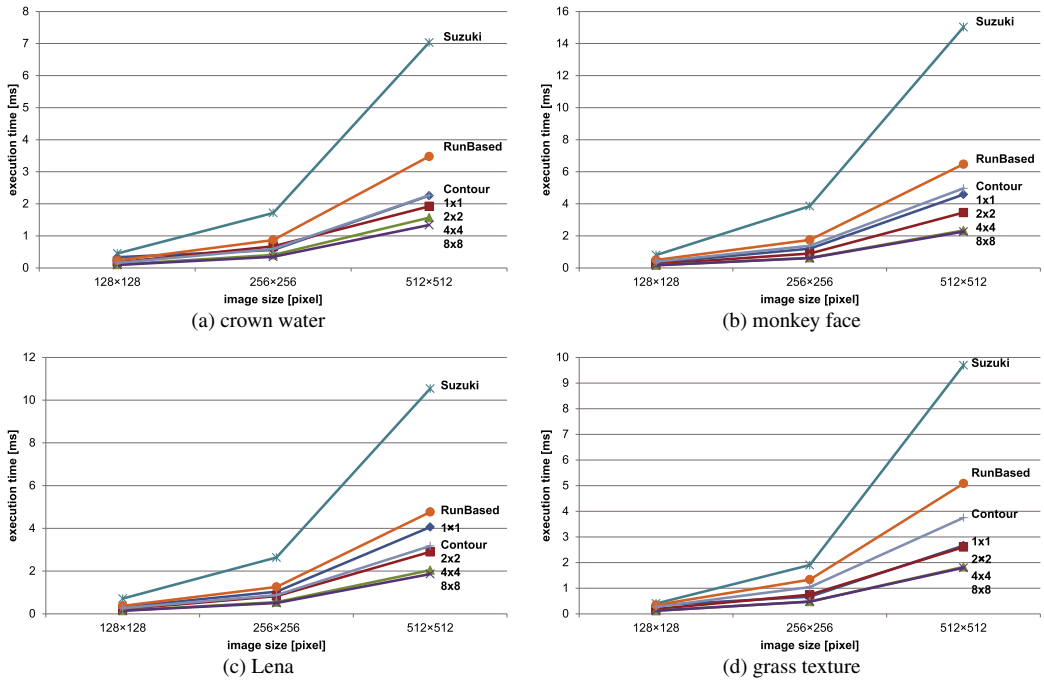
First, we evaluated execution times to obtain the 0th and 1st moment features of multiple objects in an image by software-implementing our cell-based labeling algorithm with different cell sizes on a PC. $N \times N$ pixel images (N = 128, 256, and 512) were evaluated for the four types of binary image patterns shown in Fig. 4; (a) crown water, (b) monkey face, (c) Lena, and (d) grass texture. These test images were collected from the image database at the University of Southern California[†] and the Columbia-Utrecht Reflectance and Texture Database[††], and (a) – (d) show the images of 512 × 512 pixels to be evaluated, which were binarized using Otsu method [22]. The images of different sizes to be evaluated were generated by cropping these image patterns proportionally. The cell sizes in the cell-based labeling algorithm were set to $n \times n$ pixels (n = 1, 2, 4, and 8), and the maximum number of labeled regions was set to be half the number of cells, $L = M^2/2$. In the evaluation, we used a PC with an ASUSTeK P5E mainboard, Intel Core 2 Quad Q9300 bulk CPU, 4 GB memory, and Windows XP Professional 32 bit OS. A threshold parameter θ to determine active cells was set to 1 in all the cases, and the execution times were evaluated without considering the fact that the objects in the images may be lost or degraded in the cropping. Here it is noted that the results labeled with our cell-based labeling algorithm are perfectly matched with those with conventional labeling algorithms when $n = 1$.

Figure 5 shows the execution times on the PC for our cell-based labeling algorithm for different image sizes, compared with those for conventional labeling algorithms listed in Table 1; Suzuki's algorithm [10] is a well-known four-scan algorithm for fast pixel-based connected components labeling; He's algorithm [15] is a fast raster-scan-based labeling algorithm; Chang's algorithm [12] is a fast one-scan labeling algorithm using contour tracing. In the figures, "1 × 1", "2 × 2", "4 × 4", and "8 × 8" imply execution times for connected components labeling and moment feature calculation of the labeled regions when the cell size is set to n = 1, 2, 4, and 8, respectively, in the cell-based labeling algorithm.

For the "crown water" image of 512 × 512 pixels, the execution times of cell-based labeling algorithm with n = 1, 2, 4, and 8 were 2.25 ms, 1.92 ms, 1.57 ms, and 1.34 ms, respectively, and those of Suzuki's algorithm, He's algorithm, and Chang's algorithm were 7.03 ms, 3.48 ms, and 2.26 ms, respectively. For the "monkey face" image of 512 × 512 pixels, the execution times of cell-based labeling algorithm with n = 1, 2, 4, and 8 were 4.58 ms, 3.45 ms, 2.34 ms, and 2.27 ms, respectively, and those of Suzuki's algorithm, He's algorithm, and Chang's algorithm were 15.03 ms, 6.48 ms, and 4.97 ms, respectively. For the "Lena" image of 512 × 512 pixels, the execution times of cell-based labeling algorithm with n = 1, 2, 4, and 8 were 4.06 ms, 2.90 ms, 2.04 ms, and 1.86 ms, respectively, and those of Suzuki's algorithm, He's algorithm, and Chang's algorithm were 10.54 ms, 4.77 ms, and 3.18 ms, respectively. For the "grass texture" image of 512 × 512 pixels, the execution times of cell-based labeling algorithm with n = 1, 2, 4, and 8 were 2.66 ms, 2.60 ms, 1.83 ms, and 1.80 ms, respectively, and those of Suzuki's algorithm, He's algorithm, and Chang's algorithm were 9.70 ms, 5.08 ms, and 3.75 ms, respectively. The execution times of cell-based labeling al-

[†]http://sipi.usc.edu/database/

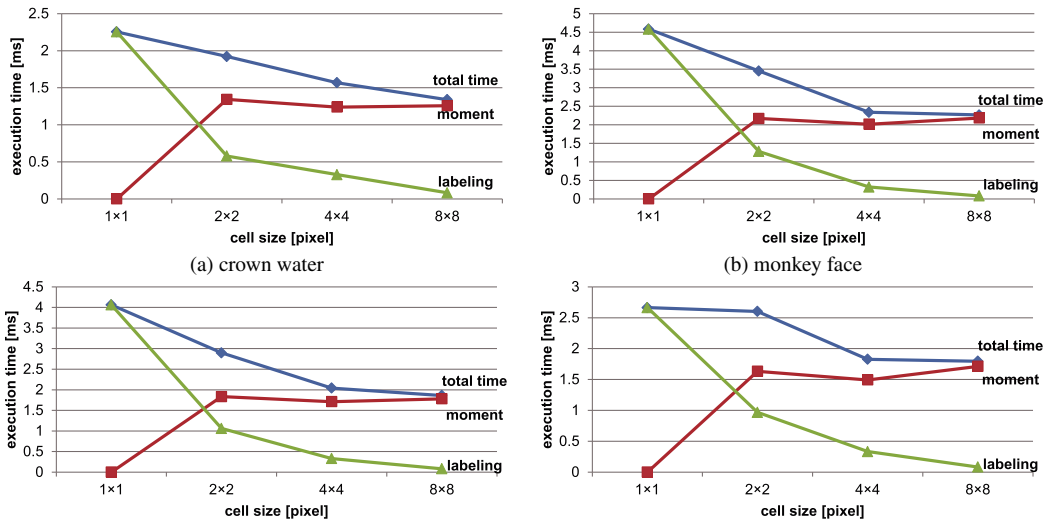
[††]http://www1.cs.columbia.edu/CAVE/software/curet/index.php



(a) crown water (b) monkey face

(c) Lena (d) grass texture

Fig. 5 Execution times for our cell-based labeling algorithm ($n = 1, 2, 4$ and 8) and conventional labeling algorithms.



(a) crown water (b) monkey face

(c) Lena (d) grass texture

Fig. 6 Relationship between execution times and cell sizes in the case of a 512×512 pixel image ($N = 512$).

Table 1 Labeling algorithms to be compared.

Algorithm	Type	Reference
Suzuki	Four-scan	Suzuki et al. [10]
Chang	Contour tracing	Chang et al. [15]
He	Run-based two-scan	He et al. [12]
CellBased	Cell-based one-scan	this paper

gorithm when $n = 1$ were less than those of the labeling algorithms to be compared for all the tested image patterns except in the case of Chang's algorithm for the "Lena" image. For all the image patterns and sizes, the execution time of cell-based labeling algorithm decreased as the cell size n increased, and the execution times of all the labeling algorithms to be compared were greater than those of cell-based labeling algorithm when $n = 2, 4$, and 8.

For the test image patterns of 512×512 pixels with the different cell sizes where $n = 1, 2, 4$, and 8, Fig. 6 shows the execution times for cell-based moment feature calculation and labeling of cells in the cell-based labeling algorithm. Here there is no need to calculate cell-based moment features when $n = 1$, because the 0th moment feature of a 1×1 cell is equal to its pixel value and pixel-level accu-

mulation for moment feature calculation is wholly involved in the process for labeling of cells. When the "Lena" image was labeled with 2×2 cells ($n = 2$), 1.84 ms and 1.06 ms were the execution time for cell-based moment feature calculation and that of labeling of cells, respectively, and its total time was 2.90 ms. Figure 6 shows that the total execution time required for cell-based labeling decreased when the cell size n increased. When the "Lena" image was labeled with 8×8 cells ($n = 8$), the execution times of cell-based moment feature calculation, labeling of cells, and their total time, were 1.78 ms, 0.08 ms, and 1.86 ms; they were 0.97 times, 0.08 times, and 0.64 times that when $n = 2$. From these figures, it can be seen that the execution time for cell-based moment feature calculation was not so largely changed with the increase of cell size n, because the computation involved in cell-based moment feature calculation is pixel-level computation of the order $O(N^2)$. On the other hand, the computation involved in labeling of cells was remarkably reduced, because its computation order of $O(M^2)$ is $1/n^2$ times that of the pixel-level computation. For labeling the other image patterns, the similar tendencies can be observed in that the execution times in cell-based labeling decreases when the cell size n increases.

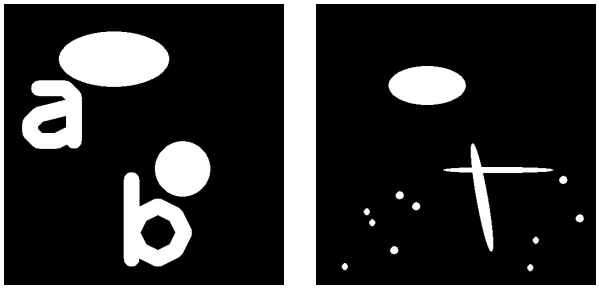
These evaluation results indicate that we can speed up the cell-based labeling algorithm especially for labeling of cells by setting a large cell size. Therefore, this implies that multi-object extraction based on cell-based labeling can be executed at a much higher speed if sufficient acceleration is provided for cell-based moment feature calculation; this requires pixel-level computation that can be easily implemented on hardware logic circuits.

4.2 Accuracy Verification

Next, we evaluated how the processed results in the cell-based labeling algorithm vary with the cell size n and threshold parameter θ to determine active cells.

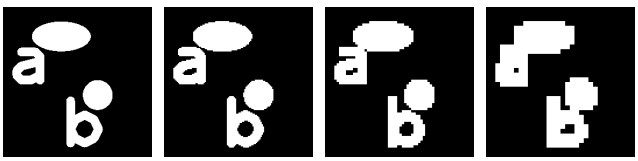
First, the 512×512 pixel binary image shown in Fig. 7 (a), in which the letters of "a" and "b", an ellipse and a circle were located, was labeled using the cell-based labeling algorithm for cell sizes of $n = 1, 2, 4, 8$, and 16. Here a threshold parameter θ was set to 1. Figure 8 shows flag maps that indicate where active cells are in $M \times M$ cells when $n = 2, 4, 8$, and 16. The flag map when $n = 1$ corresponded to the image shown in Fig. 7 (a). In the evaluation, 4, 4, 4, 3, and 2 labeled regions were counted when $n = 1, 2, 4, 8$, and 16, respectively. This is because the letter "a" and the neighboring ellipse in the image were unified in the same labeled region when $n = 8$, and the letter "b" and the neighboring circle were unified when $n = 16$. These unifications of neighbor connected components in the image can be also observed in the flag maps in Fig. 8.

Figure 9 shows the bargraphs of cell-based 0th moment features when $n = 1, 2, 4, 8$, and 16. When $n = 2$ and 4, there were no neighbor connected components and all the 0th moment features for the labeled regions were exactly matched with those when $n = 1$, corresponding to the 0th moment features calculated by using other pixel-based connected component labeling algorithms. Even when $n = 8$ and 16 with unification of the neighbor connected components, it was confirmed that all the 0th moment features were exactly matched with the added values of those of the two unified connected components when $n = 1$. The same tendency was also confirmed when calculating the 1st moment features for the labeled regions. These evaluation results suggest that the cell size n determines the separability for neighbor connected components in cell-based labeling, whereas the 0th and 1st moment features are exactly preserved by setting a threshold parameter $\theta = 1$ when there are no unified labeled regions in an image. Our cell-based labeling algorithm is not equivalent to conventional pixel-level labeling algorithms, and the same label is often given to different but close connected components because our algorithm only considers the connectivity at cell level and does not consider that at pixel level. Here cell-based labeling is equivalent to pixel-level labeling when $|x_i - x_j| > \sqrt{5}n(^\forall x_i \in O_i, ^\forall x_j \in O_j)$ for all the connected components $O_i, O_j (i \neq j)$ in an image. In



(a) image pattern 1 (b) image pattern 2

Fig. 7 Image patterns used in accuracy verification.



(a) $n = 2$ (b) $n = 4$ (c) $n = 8$ (d) $n = 16$

Fig. 8 Flag maps of image pattern 1 to indicate active cells for labeling.

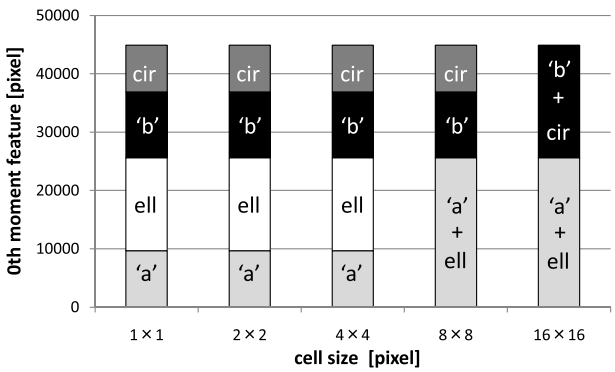


Fig. 9 0th moment features of labeled regions ("a" = the letter "a", "b" = the letter "b", ell = ellipse, cir = circle).

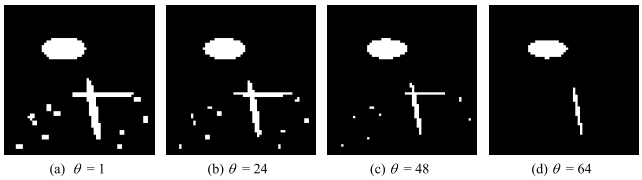


Fig. 10 Flag maps of image pattern 2 to indicate active cells for labeling.

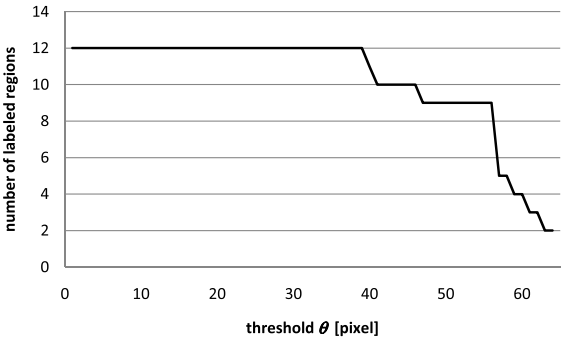


Fig. 11 Relationship between number of labeled regions and threshold parameter θ.

this case, the cell-based moment features for multiple objects extracted by cell-based labeling, are also equivalent to their moment features extracted in pixel-level labeling, because no connected component is mislabeled when the nearest distance of two different connected pixels in an image is greater than $\sqrt{5}n$.

Next, the 512×512 pixel binary image in Fig. 7 (b) was labeled by using the cell-based labeling algorithm with a cell size of 8×8 pixels ($n = 8$). The image contained a large ellipse, ten small isolated circles, and a crossed-shape object composed by two short-width ellipses. In the evaluation, a threshold parameter θ was varied from 1 to 64. Here $\theta = 1$ implies that a cell is active for labeling if there exists any active pixel in the cell, and $\theta = 64$ implies that a cell is active only when all the pixels in the cell are active. Figure 10 shows the flag maps for labeling when $\theta = 1, 24, 48$, and 64. Figure 11 shows the counted number of labeled regions in varying θ from 1 to 64. It can be observed that the number of labeled regions began to decrease at $\theta = 40$ and only two labeled regions remained for the large-sized objects at $\theta = 64$, corresponding to the disappearances of the isolated small circles in the flag maps for labeling as shown in Fig. 10. These disappearances in the flag maps imply that the small circles had no active cells, in which all the pixels were active. In Fig. 10 (d) at $\theta = 64$, the short-width ellipse in horizontal direction, which is a part of the crossed-shape object, was disappeared in the flag maps as well as the small circles. Therefore, small or narrow connected components caused by noises can be removed by adjusting a threshold parameter θ, and this implies that we can control robustness and sensitivity to extract multiple objects in an image in cell-based labeling.

Figure 12 shows the 0th moment features and deviations from their true centroid positions for the ellipse and crossed-shape object in the image, varying with θ from 1 to 64. Here the 0th moment feature of the labeled region corresponds to its area. The centroid positions were calculated by dividing the 1st moment features with the 0th moment feature as shown in Eq. (23), and the true centroid positions were assumed as those when $\theta = 1$, which were exactly matched with those calculated using pixel-based connected components labeling algorithms. In Fig. 12, the 0th moment feature of the ellipse decreased when θ increased, whereas there was little deviation in the centroid position of the ellipse even when θ was large. For the crossed-shape object, the decreasing degree of the 0th moment feature and the deviation of its centroid position became much greater than those for the ellipse when θ increased. This is because the narrow and asymmetric shape of the crossed-shape object was strongly reduced in the flag map for labeling by setting θ to a large value as shown in Fig. 10.

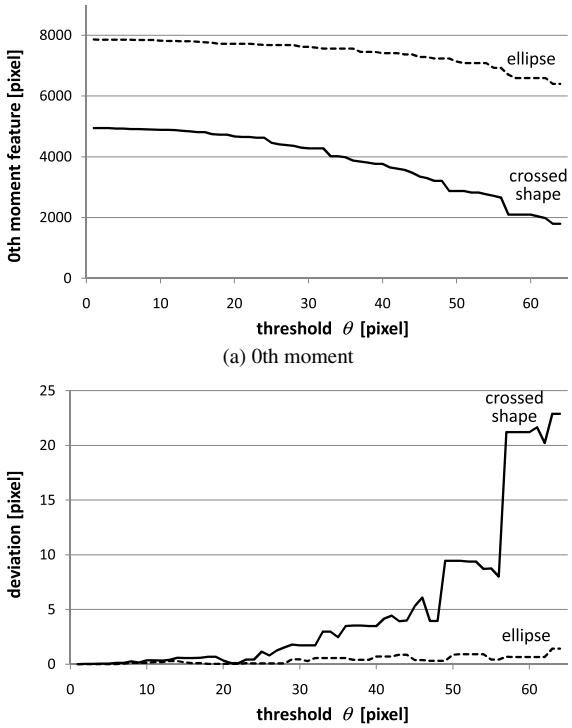


Fig. 12 0th moment features and deviations from true centroid positions.

Therefore, when we adjust a threshold parameter θ to determine the active cells in the cell-based labeling algorithm, it can be seen that there is a trade-off relationship between robustness to small and narrow connected components in an image and accuracy in moment feature calculation for obtaining the positions and sizes of the labeled regions.

4.3 Memory Consumption

Finally, we considered how the memory consumption for moment features in executing the cell-based labeling algo-

Table 2 Numbers and bit widths of labels in the case of a 512×512-pixel 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 3 Memory consumption in the case of a 512 × 512-pixel image.

cell size n	(1) QL	(2) Mw	(3) Lw	total
1	1,163,264	1,088	278,528	1,442,880
2	290,816	480	61,440	352,736
4	72,704	208	13,312	86,224
8	18,176	88	2,816	21,080

unit: byte = 8 bits

rithm varies with the cell size n and the maximum number of labels L. For a binary image of $N \times N$ pixels, the 0th moment feature consumes $\lceil \log_2 N^2 \rceil + 1$ bits, and the two 1st moment features consume $2\left\lceil \log_2 \frac{N(N-1)(N-2)}{2} \right\rceil$ bits;

$$Q = (\lceil \log_2 N^2 \rceil + 1) + 2\left\lceil \log_2 \frac{N(N-1)(N-2)}{2} \right\rceil, \quad (25)$$

where $\lceil x \rceil$ denotes the ceiling function which gives the smallest integer $\geq x$. When there are L connected components to be labeled in the image, the number of bits needed to express their label numbers is an unsigned integer of $w = \lceil \log_2 L \rceil$ bits. In this study, (1) QL bits for the 0th and 1st moment features 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 in executing our cell-based labeling algorithm with $M \times M$ cells of $n \times n$ pixels. For the maximum number of labeled regions set at $L = M^2/2$, Table 2 shows L and w when a 512×512-pixel image ($N = 512$) is processed with cell sizes of $n = 1, 2, 4$, and 8. Table 3 shows the memory consumption when a 512×512-pixel image is processed with cell sizes of $n = 1, 2, 4$, and 8. Here, the 0th and 1st moment features consume $Q = 71$ bits for a labeled region in a 512×512-pixel image. When $n = 1$, (1) the moment features of L labeled regions, (2) the label numbers of M cells, and (3) the tentative memories in the relabeling process occupy 1,163,264 bytes, 1,088 bytes, and 278,528 bytes, respectively. The total memories consumed are 352,736 bytes, 86,224 bytes, and 21,080 bytes, respectively, when $n = 2, 4$, and 8; these values are 0.244 times, 0.060 times, and 0.015 times that of 1,442,880 bytes when $n = 1$. In Table 3, it can be observed that our cell-based labeling algorithm can greatly reduce memory consumption in multi-object extraction with a larger cell size.

Table 4 shows the total memory consumption in executing the cell-based labeling algorithm with different cell sizes for different image sizes when $L = M^2/2$. It can be observed that the memory consumption increases as the image size N becomes larger, and there are similar tendencies in which the total memories consumed are remarkably reduced

Table 4 Memory consumption in different image sizes and cell sizes.

image size	cell size $n = 1$	cell size $n = 2$	cell size $n = 4$	cell size $n = 8$	cell size $n = 16$
128 × 128	69	17	5	1	1
256 × 256	321	79	20	5	2
512 × 512	1,410	345	85	21	6
1024 × 1024	6,403	1,570	385	95	24
2048 × 2048	27,654	6,787	1,666	409	101

unit: kbyte

with a larger cell size n. The reduced memory consumption with a larger cell size is effective in hardware-implementing our algorithm for high-speed multi-object extraction.

For example, we consider memory consumption in implementing the cell-based labeling algorithm as the hardware logic in a commercial FPGA (Xilinx XC3S5000-4FG900); this FPGA is specifically designed to meet the needs of high-volume and cost-sensitive consumer electronic applications and has an embedded block RAM of 239 kbytes. When the block RAM in the FPGA can be completely used for hardware implementation, a 128×128-pixel image can be processed with the cell-based labeling algorithm when $n = 1$, i.e., pixel-level connected components labeling, whereas it is unable to implement pixel-level connected components labeling for images of 256×256 pixels or more in the FPGA because of the shortage of memory resources. For images of 256×256 pixels, 512×512 pixels, 1024×1024 pixels, and 2048×2048 pixels, the minimum cell sizes that enable hardware implementation of the cell-based labeling algorithm in the FPGA are $n = 2$, 4, 8, and 16, respectively. In fact, Gu et al. [24] have hardware-implemented the cell-based moment feature calculation in our cell-based labeling algorithm with a cell size of $n = 8$ in a user-specific Xilinx XC3S5000 FPGA on a high-speed vision platform, and several experiments involving real-time multi-object extraction have been performed at a frame rate of 2000 fps.

From these considerations, it can be confirmed that our cell-based labeling algorithm, with a larger cell size, can be implemented without consuming large amounts of memory resources, even when higher-resolution images are processed; this enables hardware implementation of high-speed multi-object extraction in an FPGA with a limited memory resource.

5. Conclusion

We introduced a cell-based connected components labeling algorithm for fast multi-object extraction that can calculate the 0th and 1st moment features of the labeled regions in an image. We also verified its execution time and accuracy on a PC, and compared these with those of conventional connected components labeling algorithms. Although our cell-based labeling algorithm is not equivalent to conventional labeling algorithms when the cell size is larger than 1×1 pixels, it can be accelerated for high-resolution images without consuming a large amount of memory by adjusting the cell size and number of labels. Although labeling of cells

that requires cell-level computation is not suitable for parallel implementation, it can be easily accelerated by cell-based labeling with larger cell sizes. Calculation of cell-based moment features requires pixel-level computation, and it can be accelerated using hardware logic in an FPGA because this process is suitable for parallel implementation. The concept of cell-based labeling in this study can be applied not only to moment features but also to various types of image features with additivity for pattern recognition, such as color histograms and higher local autocorrelation features. On the basis of our results, we will extend the applications of this concept to high-speed and intelligent blob-processing in various fields such as factory automation, and multi-object target tracking and motion capture technologies in multimedia and robot control; our concept of cell-based labeling will be effective especially when real-time processing at high speed is strongly requested rather than pixel-level accuracy.

References

[1] 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.
[2] 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. Very Large Scale Integr (VLSI) Syst., vol.4, no.3, pp.322–335, 1996.
[3] T. Komuro, S. Kagami, and M. Ishikawa, "A dynamically reconfigurable SIMD processor for a vision chip," IEEE J. Solid-State Circuits, vol.39, no.1, pp.265–268, 2004.
[4] I. Ishii, K. Yamamoto and M. Kubozono, "Higher order autocorrelation vision chip," IEEE Trans. Electron Devices, vol.53, no.8, pp.1797–1804, 2006.
[5] 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.
[6] 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.
[7] I. Ishii, T. Taniguchi, R. Sukenobe, 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.
[8] A. Rosenfeld and J.L. Pfaltz, "Sequential operation in digital picture processing," J. Assoc. Comput. Mach., vol.13, pp.471–494, 1966.
[9] R.M. Haralick, "Some neighborhood operations," in Real Time/Parallel Computing, Image Analysis, eds. M. Onoe, K. Pres-ton, A. Rosenfled, pp.11–35, Plenum Press, New York, 1981.
[10] 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.
[11] K. Wu, E. Otoo, and K. Suzuki, "Two stategies to speed up connected component labeling algorithms," Tech. Rep. 59102, Lawrence Berkeley National Lab, 2005.
[12] 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.
[13] L. He, Y. Chao, K. Suzuki, and K. Wu, "Fast connected-component labeling," Pattern Recognit., vol.42, pp.1977–1987, 2008.
[14] E. Mandler and M.F. Oberlander, "One-pass encoding of connected components in multi-valued images," Proc. IEEE Conf. Comp. Vis. Patt. Recog., pp.64–69, 1990.
[15] 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.
[16] D.G. Bailey and C.T. Johnston, "Single pass connected components analysis," Proc. Image and Vis. Comput., pp.282–287, 2007.
[17] C.J. Nicol, "A systolic approach for real time connected component labeling," Comput. Vis. Image Underst., vol.61, pp.17–31, 1995.
[18] A. Baumker and W. Dittrich, "A new parallel MIMD connected component labeling algorithm," Proc. Int. Parallel Process. Symp., pp.429–433, 1996.
[19] V. Chaudhary and J.K. Aggarwal, "Parallel image component labeling for target acquisition," Opt. Eng., vol.37, no.7, pp.2078–2090, 1998.
[20] A. Amir, L. Zimet, A. Sangiovanni-Vincentelli, and S. Kao, "An embedded system for an eye-detection sensor," Comput. Vis. Image Underst., vol.98, no.1, pp.104–123, 2005.
[21] P. Gabbur, H. Hua, and K. Barnard, "A fast connected components labeling algorithm and its application to real-time pupil detection," Mach. Vis. and Appl., pp.779–787, 2009.
[22] N. Otsu, A threshold selection method from gray-level histograms, IEEE Trans. Syst. Man Cybern., vol.9, no.1, pp.62–66, 1979.
[23] 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.
[24] 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.

Qingyi Gu received the B.E. degree in Electronic and Information Engineering from Xi'an Jiaotong University, China, in 2005. He received the M.E. degree in Engineering, Hiroshima University, Japan, in 2010. He is currently a Ph.D. student in Graduate School of Engineering, Hiroshima University, Japan. His primary research interest is high-speed image feature extraction.

Takeshi Takaki received the B.E. degree and M.E. degree in Mechanical Engineering from Tokyo University of Science, Japan, in 2000 and 2002, respectively. He received the Ph.D. degree in Mechano-Micro Engineering from Tokyo Institute of Technology, Japan, in 2006. He is currently an associate professor in Graduate School of Engineering, Hiroshima University, Japan. His general research interests are robot hand, continuously variable transmission and force visualization mechanism.

Idaku Ishii received the B.E. degree, M.E. degree and Ph.D. degree from the University of Tokyo, Japan, in 1992, 1994, and 2000 respectively. He is currently a professor in Graduate School of Engineering, Hiroshima University, Japan. His general research interests are high-speed robot vision, sensory information processing, sensor-based robot manipulation, and applications in industry and biomedicine.