2010 International Symposium on Nonlinear Theory and its Applications
NOLTA2010, Krakow, Poland, September 5-8, 2010

Nonlinear Image Processing for Multiple Object Tracking on Cellular Hardware Platform

Takao Matsui †, Tomohiro Fujita †, Mamoru Nakanishi ‡, Takeshi Ogura †

† Ritsumeikan University, 1–1–1 Noji-Higashi, Kusatsu, 525–8577
† E-mail: ri008067@ed.ritsumei.ac.jp, {tfujita, togura}@se.ritsumei.ac.jp
‡ NTT Microsystem Integration Laboratories, 3–1, Morinosato Wakamiya, Atsugi, 243–0198
‡ E-mail: mamoru@aecl.ntt.co.jp

Abstract- We developed a nonlinear image processing method for multiple object tracking on cellular automata on content addressable memory (CAM^2), which is a cellular hardware platform that can attain pixel-order parallelism on a single PC board. We successfully implemented edge detection, hole filling, and center point detection, all of which are based on Cellular Automata (CA) processing, on CAM^2. Evaluation results show that real-time multiple object tracking can be performed on CAM^2 and that its performance is superior to that of conventional methods.

1. Introduction

Multiple object tracking is one of the most useful image processing applications in the security field. For multiple object tracking, various image processes—such as noise reduction, edge detection, hole filling, and center point detection—are necessary. Nonlinear image processing techniques, based on morphology and discrete-time cellular neural networks (DTCNN) processing have already been proposed [1] on a cellular hardware platform "called cellular automata on content addressable memory" (CAM^2).

On CAM^2, each cell is connected to the nearest neighbor cells and all cell values can simultaneously change by referring to the values of these neighbors. This makes CAM^2 a good fit for Cellular Automata (CA)-based techniques, such as morphology and DTCNN [2][3][4], but although DTCNN has already been successfully implemented on CAM^2, its performance is inadequate in terms of processing time and program code size.

We propose CA-based image processing for multiple object tracking and implementation on CAM^2 in place of DTCNN. Our method performs edge detection, hole filling, and center point detection at a lower processing time than DTCNN processing, demonstrating its potential for multiple object tracking at high speeds.

2. Cellular Hardware Platform: CAM^2

2.1. CAM^2 Features

CAM^2 was established on Content Addressable Memory (CAM) and CAM-based system technologies five years ago. At that time a highly parallel integrated circuits and system (HiPIC) [5] was proposed as a CAM-based system model for real-time image processing. The HiPIC is an application-specific system that achieves high performance and flexibility, and we have developed various practical real-time image processing and cellular type processing systems [6] [7] based on it. Figure 1 shows a block diagram of CAM^2.

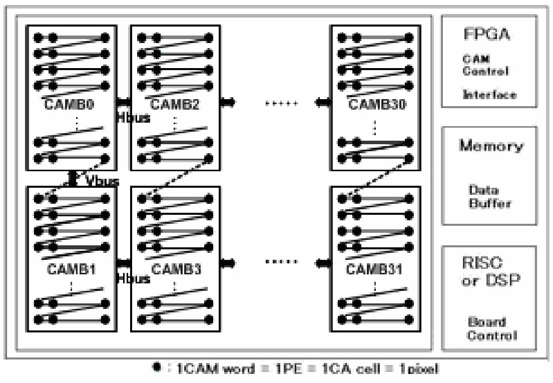

Fig. 1. Block diagram of CAM^2.

According to the HiPIC concept, CAM^2 consists of a highly parallel PE array, an FPGA that controls the array, a RISC processor or DSP that executes serial data processing, and a specified amount of memory. Each is implemented using only digital LSI technology. The most prominent features of the configuration are dedicated CAMs for the highly parallel PE array. Each CAM performs various types of parallel data processing for CA in each word. Since the memory-based structure of the CAM is the most suitable for

LSI technology, several hundred thousand CA cells (or "processing elements") can be created on a single PC board using deep sub-micron CMOS technology. Moreover, CAM^2 can be easily controlled by command sequences generated from the FPGA, which is a reconfigurable logic element. In short, CAM^2 can effectively perform CA-based image processing.

2.2. Cellular Automaton Processing on CAM^2

CA processing using CAM^2 is carried out by iterative operations of CA-value transfer and update. In the CA-value transfer process, the values of an original cell are transferred to the nearest neighbor's cell. In the CA-value update process, the next value of the original cell is calculated by a transition rule using the original and nearest neighbors' cell values.

To perform this processing, the following functions are absolutely essential:

- Maskable OR search
- Partial & parallel write
- Hit-flag shift up & down

For the search, the hit results are accumulated into hit-flag registers by means of OR logic. For the write, the data are written into specific bit positions of multiple word locations. For the shift, the hit-flag registers are shifted to up or down words. Through the iteration of there functions, CA-value transfers and updates can be performed in a bit-serial, word-parallel manner. The drawback is that the processing takes too long for complex operations such as the multiplication of longer bits. Moreover, the processing time for CA-value transfer is in proportion to the transfer bit length. Thus, low-bit CA-value updates are required to shorten the processing time.

3. Morphology and DTCNN Processing for Multiple Object Tracking

Morphology and DTCNN processing have already been proposed for multiple object tracking [1]. An example of image processing for multiple object tracking using CAM^2 is shown in Fig2. In the first step of this process, object outlines are extracted by edge detection performed by DTCNN processing. Next, hole filling is performed by DTCNN processing followed by noise reduction perfomed by dilation of morphology. Finally, center points of the objects are extracted by DTCNN processing.

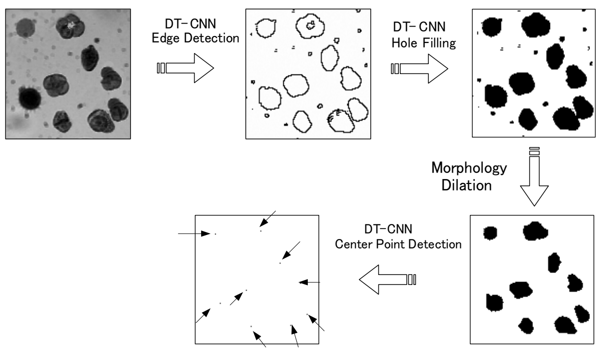

Fig. 2. Example of image processing.

4. CA-based Processing on CAM^2 for Multiple Object Tracking

4.1. CA-based Edge Detection

In CA-based edge detection, the transition rule for the CA-value update is quite simple, only when all the nearest neighbor cells are black should the next CA-value be changed (form black to white), as shown in Fig. 3. The results of CA-based and DTCNN-based edge detection are the same.

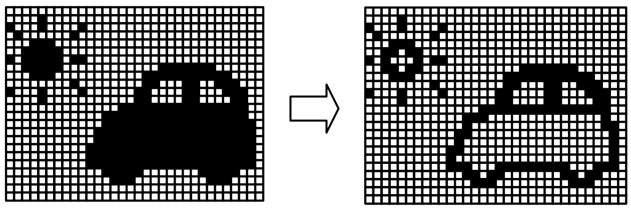

Fig. 3. Example of CA-based edge detection.

4.2. CA-based Hole Filling

In CA-based hole filling, the output image is generated using the original image and a transition rule. The original image is pre-served through the processing, and the initial output image is all black, as shown in Fig. 4(a). The output image is then moderated in a wave propagation manner by the CA-value update, as shown in Fig. 4 (b) to (f). This update is repeated iteratively based on the pixel size. The transition rule is quite complicated, and the process converges after the pixel size steps.

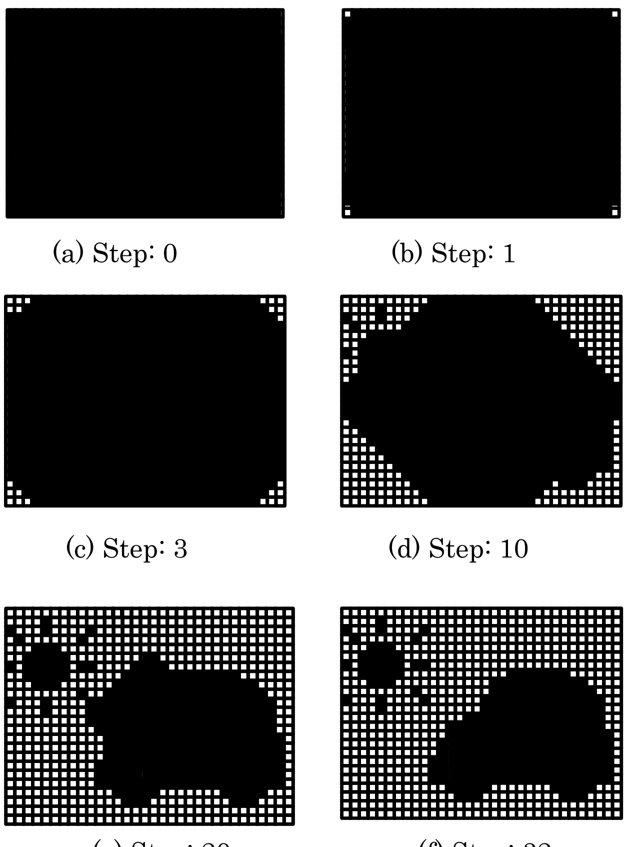
(a) Step: 0 (b) Step: 1
(c) Step: 3 (d) Step: 10
(e) Step: 20 (f) Step: 32

Fig. 4. Example of CA-based hole filling.

4.3. CA-based Center Point Detection

CA-based center point detection is composed of four processing elements: upper-side, lower-side, left-side, and right-side. The transition rule of the upper-side processing is shown in Fig. 5. When the upper cells are all white, and one or more lower cells are black, the center cell changes from black to white.

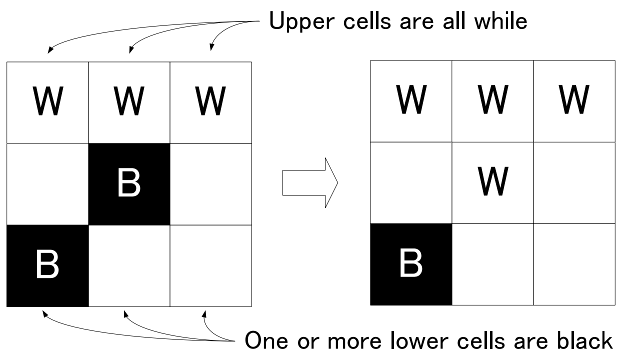

Fig. 5. Transition rule of upper-side processing

The transition rule of under-side, right-side, and left-side processing is similar to that of upper-side processing but not quite the same. The four processes are repeated iteratively until no processed cells remain. CA-based center point detection is shown in Fig. 6.

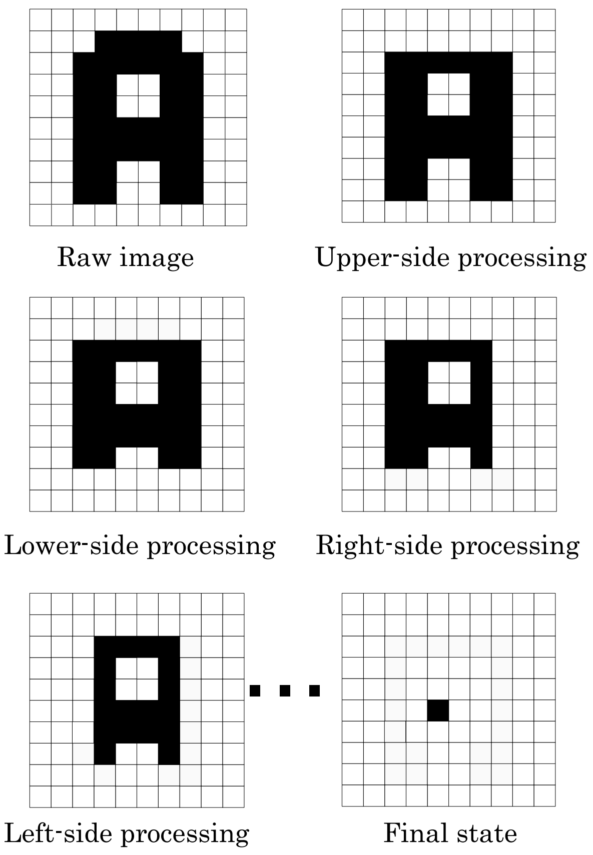

Fig. 6. Example of CA-based center point detection.

5. Experiments and Evaluation

5.1. Processing Performance

Fig. 7 shows a photograph of CAM^2. A highly parallel PE array (a CAM LSI) is mounted on a daughterboard, while a motherboard contains an FPGA and some memory. CAM^2 can handle a 128 × 128 image at 40 MHz system clock. The processing time of CAM^2 is not influenced by image size when the image size is smaller than 128 × 128.

Processing times are summarized in Tables 1, 2, and 3. CA-based edge detection is about 12 times faster than that with DTCNN processing, CA-based hole filling is about 40 times faster, and CA-based center point detection is about 13 times faster. In total, CA-based processing is 16 times faster than DTCNN processing.

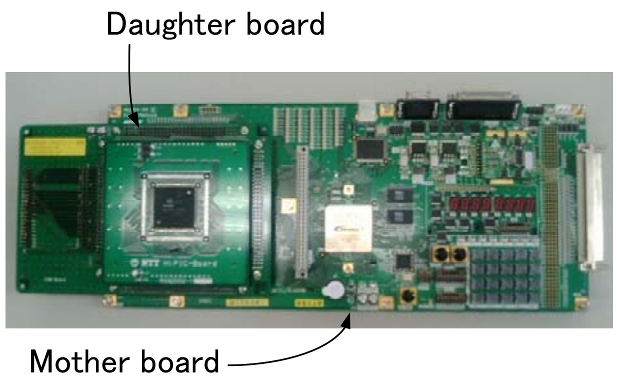

Fig. 7. CAM^2 board with 128 ×128 CA cells.

Table 1.Edge detection processing time

	Number of cycles	Time (s)
DTCNN	2350	58.750μ
CA-based	189	4.725μ

Table 2. Hole filling processing time

	Number of cycles	Time (s)
DTCNN	162305	4.0576 m
CA-based	4416	0.1104 m

Table 3. Center point detection processing time

	Number of cycles	Time (s)
DTCNN	458008	11.450 m
CA-based	3536	0.8840m

5.2. Example of CA-based Image Processing

Fig. 8 shows an example of CA-based image processing for multiple object tracking. Various types of objects are processed, and finally the object center points are successfully extracted. This result is the same as the one achieved with DTCNN processing.

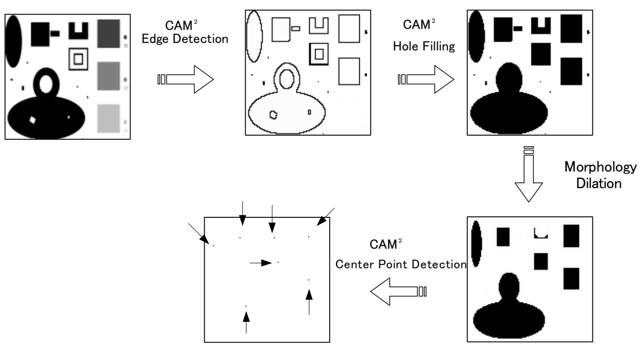

Fig. 8. Example of CA-based image processing.

6. Conclusion

We described a CA-based image processing technique on CAM^2 for multiple object tracking. In place of DTCNN processing, CA-based edge detection, hole filling, and center point detection are proposed and successfully implemented on CAM^2. Experimental results demonstrated that CA-based processing takes less time than DTCNN processing. To develop a multiple object tracking system for actual use, we plan to study post-processing techniques, such as judgment for correspondence between objects, for use with the CA-based algorithm.

References

[1] Takeshi Ikenaga and Takeshi Ogura, "Real-time Morphology Processing Using Highly-parallel 2D Cellular Automata CAM^2," IEEE Trans. Image Processing, Vol. 9, No. 12, pp. 2018–2026, 2000.

[2] Takeshi Ikenaga and Takeshi Ogura, "Discrete-time cellular neural network using a highly-parallel 2-D cellular automata CAM^2," Proc. NOLTA'96, 1996,pp.221–224.

[3] Takeshi Ikenaga and Takeshi Ogura, "A DTCNN universal machine based on highly parallel 2-D cellular automata CAM^2," IEEE Trans. Circuits Syst. I, Vol. 45, No. 5 pp.538–546, 1998.

[4] Fujita Tomohiro, Nakanishi Mamoru, and Ogura Takeshi, "CAM^2 -Universal Machine: A DTCNN Implementation for Real-Time Image Processing," Proc. of 11th International Work-shop on Cellular Neural Networks and their Applications, pp. 219–222,2008.

[5] Takeshi Ogura and Mamoru Nakanishi, "CAM-Based Highly-Parallel Image Processing Hardware," IEICE Trans. Electtron., Vol. E80-C, No. 7, pp. 868–874, 1996.

[6] M. Nakanishi and T. Ogura, "Real-time CAM-based Hough Transform Algorithm and Its Performance Evaluation," J. of Machine Vision and applications, Vol. 12, No. 2, pp. 59–68, 2000.

[7] Kazuhito Sakomizu, Tomohiro Fujita, Jyunji Ueda, Takashi Okamura, Kota Minoura, and Takeshi Ogura, "Probabilistic Information processing on CAM^2 with Mean Field Approximation," J. of Signal Processing, Vol. 12, No. 4, pp. 279–282, 2008.