A Method Using CA-Based Pixel-Level Snakes for Multiple-Contour Expansion

Sho Sawada¹, Tomohiro Fujita¹, Kaihei Iwanaga², Takeshi Kumaki¹, Mamoru Nakanishi² and Takeshi Ogura¹

¹ Ritsumeikan University
1-1-1 Noji-higashi, Kusatsu, Shiga 525-8577, Japan
{ri0003pv@ed, tfujita@se}.ritsumei.ac.jp
{kumaki@fc, togura@se}.ritsumei.ac.jp

² NTT Microsystem Integration Laboratories
3–1 Wakamiya, Morinosato, Atsugi, Kanagawa 243-0198, Japan
mamoru@aecl.ntt.co.jp

Abstract

One of the variations of the active-contour method, namely, multiple-contour expansion, is realized on cellular automaton (CA)-based pixel-level snakes (CA-based PLS). As for the PLS algorithm, the active contour steps to next pixel one by one. The processing speed of PLS thus depends on image size. In the case of the multiple-contour-expansion method, the contours are successively deformed, their topology is transformed, and objects are extracted from images. The advantage of this method is that the processing speed does not depend on image size. It was experimentally demonstrated that multiple contour expansion is made possible by the CA-based PLS and that the processing speed of PLS does not depend on image size.

1. Introduction

Object extraction is one of key issues concerning the discipline known as computer vision. The active contour method (or also known as the “snake” algorithm) is one of the methods for object extraction [1]. “Pixel-level snakes” (PLS) [2–5] is a kind of snake algorithm. As for PLS, the development of the active contours is governed by a balance of two forces: one from an internal energy of the active contours and one from an external potential of a processing image. This algorithm is based on cellular neural networks (CNN), and so it can be processed in parallel and implemented on parallel-processing machines. In a previous work [6], we implemented PLS on a parallel-processing machine, CAM² [7]. In spite of our effort, the processing speed of our PLS is, however, far from that required for real-time image processing.

In this study, a new approach, namely, cellular automaton (CA)-based PLS, is proposed. This CA-based approach involves a series of simple CA rules. In previous work [6], the bottleneck part in some process was the implementation of CNN on CAM². CAM² was developed as a CA-dedicated hardware engine, so CA works faster than CNN on CAM². Real-time image processing is expected to be possible by taking a CA-based approach.

Since the number of process iterations in CA-based PLS depends on image size, the processing speed of CA-based PLS also depends on image size. To eliminate this dependency, a multiple-contour expansion method is proposed here. As for this method, initial multiple contours are successively deformed, their topologies are transformed, and objects in images are extracted. The number of process iterations only depends on the number of pixels between adjacent contours. As a result, in the case of the proposed CA-based algorithm, processing speed does not depend on image size.

This paper is structured as follows. In Sec. 2., our proposed method, CA-based PLS algorithm, is proposed. The multiple-contour-expansion method is proposed in Sec. 3.. Experimental measurements of the CA-based PLS are presented in Sec. 4.. Finally, this paper is concluded in Sec. 5..
Table 1: Label assignment to CA state

| Label | Assigned state          |
|-------|-------------------------|
|       | outer cell              |
| ■     | contour cell            |
| □     | inner cell              |
| 3     | outer or contour cell   |
| 4     | outer or inner cell     |
| 5     | contour or inner cell   |
| 6     | all kinds of cell       |

takes much more time. Real-time image processing was thus not achieved by our original CNN-based PLS algorithm. From this reason, we propose a CA-based PLS algorithm for processing on CAM$^2$. The proposed CA-based PLS uses only basic operations of CAM$^2$, so it can be executed at high speed. Owing to parallel processing of CAM$^2$, all the CA rules can be processed simultaneously at all pixels. From this reason, it can be considered that the CA-based PLS is suitable for CAM$^2$.

The proposed CA-based PLS algorithm is realized by three-state CA. The three states correspond to the three types of cells, and first, second, and third states are assigned to the cells on contours, the cells in the inner region, and the cells in the outer region, respectively. The motion of a closed curve is determined at the pixel level by dominant force between two forces caused by internal energy and external potential.

The conceptual scheme of the CA-based PLS is depicted in Fig. 1. In the figure, the value of $F_{\text{int}}$ is given by CA rules. $F_{\text{int}}$ is compared with force $F_{\text{ext}}$ caused by the external potential, and if $F_{\text{int}}$ is greater than $F_{\text{ext}}$, the center cell in the figure changes into a contour cell. $F_{\text{ext}}$ was calculated by the absolute value of the gradient of the pixel value.

The CA rules for $F_{\text{int}}$ are proposed as follows. A rule that gives $F_{\text{int}}$ in the contour-expansion case is proposed. The contour-contract version of CA rules can also be realized by swapping the assignments of the outer and inner cells in the expansion’s rules.

2.1 Forces in vertical and horizontal directions

The rules of the forces by the internal energy in the vertical and horizontal directions are shown in Fig. 2 (a)–(c). In Fig. 2, the labels of the cells mean the states of the cells. The labels are assigned to the states as shown in Tab. 1. Figures 2 (a)–(c) only show the north direction, and the other directions can be obtained by rotating these rules. Figure 2 (a) shows the force caused by the membrane energy. This force represents the contractile force that resists the stretch of the contour. A weak force ($F_{\text{int}} = 1$) is assigned to this kind of force. Figures 2 (b) and (c) show the force caused by the thin plate energy, which represents the resisting force against the bending. The bending in Fig. 2 (c) is sharp, thus the force $F_{\text{int}}$ caused by (c) is stronger than one by (b).

2.2 Forces in diagonal direction

The CA rules for the force in the northwest direction is shown in Fig. 2 (d). The meanings of the labels are listed in Tab. 1. The forces of the other diagonal direction can be obtained by rotating this rule. This force is caused by the membrane energy.

2.3 Thinning rule

When the contour evolves by the forces, the line width of the contour increases. The unnecessary contour must be removed to keep the line thin. A thinning rule is shown in Fig. 3. As shown in the figure, the contour cell that is not next to the inner cell transits to the outer cell.

2.4 Connectivity rule

In the process of PLS algorithm, continuity of the contour pixels must be guaranteed. For this purpose, if a discontinuous point is encountered, the algorithm has to modify it to maintain the connectivity of the contour. The rule for connectivity is shown in Fig. 4. This rule states that if an outer cell is next to an inner cell in the horizontal or vertical direction, that outer cell is replaced with a contour cell.
3. Multiple-Contour-Expansion Method

The proposed multiple-contour-expansion method is described as follows. The processing speed of CA-based PLS depends on image size. The multiple contour expansion method eliminates this dependency and realizes high-speed image processing.

As mentioned in the previous section, CA-based PLS is suitable for CAM². However, its application still faces some problems that must be overcome before real-time processing is possible. One problem is speeding up processing for large-scale images. When a single initial active contour starts from a size that is too small compared with the size of the object, the transverse distance of the active contour becomes too long. An active contour shifts by only one pixel in every CA process. So, in this case, extraction of object contours takes much time. Large-scale images have many more pixels, so the speed of this one-pixel-by-one-pixel algorithm depends on image size.

To eliminate this image-size dependency, the multiple-contour-expansion method is proposed. In the case of this method, multiple contours are placed in an initial condition and then expanded. When the active contour collides with the adjacent contours, these contours are merged into one, and their topologies are transformed. The conceptual scheme of this topology transformation is depicted in Fig. 5.

Figure 5 shows two contours collide. The contour cells are not next to inner cells at the collision point. The thinning rule is applied to these cells, and they change into outer cells. After this process, these two contours are merged, and the topology is transformed.

Active contours are enlarged by the above-described contour-merging process. This enlargement process is depicted in Fig. 6, where the force from the external potential is assumed to be zero. Figure 6 (a) shows multiple active contours arranged in an array. In the first step, these active contours expand and collide with each other (Fig. 6 (b)). In the second step, the topology transformation takes place, and multiple active contours are merged into one large active contour. If multiple active contours are arranged in the object contour, the merged active contour is effectively enlarged in a few steps. So the enlargement with the contour-merging process is not a one-pixel-by-one-pixel process.

The computational effort required by the multiple-contour-expansion method is equal to that required in the single-contour case, because, in the cases of both methods, the same CA rules are applied. Owing to parallelism of CAM², all CA rules are simultaneously processed at all pixels. For these two reasons, namely, being the CA-based PLS algorithm is not a one-pixel-by-one-pixel process and parallelism of CAM², the number of process iterations does not depend on the size of processing images.

The procedure of the multiple-contour-expansion method is explained hereafter. Multiple contours are installed in an array in the initial state. If the initial active contour is located on the object contour, the final contour is broken at this point. The object contour has a large gradient in pixel brightness. Therefore, the initial contour is not located at such points. The rules proposed in the previous section are applied to these multiple contours. Multiple contours repeat expansion and topology transformations, and the active contours terminate at object edges.

4. Experiments

The results of experiments on multiple-contour expansion are shown in Fig. 7. In Fig. 7, the objects are extracted. According to Figure 7, the proposed multiple-contour-expansion method performs topological transformation well. That is, our CA rules work correctly.

The image-size dependency of the multiple-contour-
expansion algorithm was investigated as follows. The processing speeds of the conventional method, that is, a single-contour contract method, and the multiple-contour-expansion method are compared below.

The number of the process iterations required by both methods is shown in Fig. 8. The image with a size of is 640 × 480 is the original. The images with sizes of 480 × 360 or 960 × 720 are obtained by shrinkage or enlargement of the original image, respectively, in Fig. 8. According to this figure, the processing speed of the single-contour-contract method depends on the image size; on the other hand, that of the multiple-contour-expansion method does not depend on image size.

In the case of single-contour contraction, many more pixels exist between the active contour and the object edge. That results in many more iterations of the algorithm. In contrast, in the case of multiple-contour expansion, the number of iterations depends on the number of pixels between two adjacent active contours, so the image-size dependency disappears.

5. Conclusion

A multiple-contour-expansion method with CA-based PLS was proposed. The results of an experiment on multiple-contour expansion show the CA-based PLS correctly extracts an object and maintains contour topology. The processing speed of CA-based PLS does not depend on number of contours. The efficiency of the multiple-contour-expansion method is expected to increase when much smaller contours are started with, so the proposed method increases the processing speed of object extraction. In the next step of this study, this method will be applied to many more images in order to check its robustness. Moreover, the method should be compared with the original version of PLS from the viewpoints of extraction performance and processing speed. In addition, the method must be implemented on CAM$^2$ as future work. Only then we will be able to evaluate the possibility of real-time image processing.

References

[1] M. Kass, A. Witkin and D. Terzopoulos: Snakes: Active contour models, International Journal of Computer Vision, Vol. 1, pp. 321–331, 1988.
[2] D. Vilarino, D. Cabello, A. Mosquera and J. Pardo: Application of a multilayer discrete-time CNN to deformable models, in Biological and Artificial Computation: From Neuroscience to Technology, Lecture Notes in Computer Science, J. Mira, R. Moreno-Díaz and J. Cabestany, Eds., Springer Berlin Heidelberg, Vol. 1240, pp. 1193–1202, 1997.
[3] D. Vilarino, D. Cabello, X. Pardo and V. Brea: Cellular neural networks and active contours: A tool for image segmentation, Image and Vision Computing, Vol. 21, No. 2, pp. 189 – 204, 2003.
[4] D. Vilarino and C. Rekeczky: Implementation of a pixel-level snake algorithm on a CNNUM-based chip set architecture, IEEE Trans. Circuits and Systems I, Vol. 51, No. 5, pp. 885 – 891, 2004.
[5] D. L. Vilarino and C. Rekeczky: Pixel-level snakes on the CNNUM: Algorithm design, on-chip implementation and applications, Int. J. Circuit Theory Appl., Vol. 33, No. 1, pp. 17–51, 2005.
[6] T. Matsui, T. Fujita, Y. Tsuji, T. Kumaki, M. Nakanishi and T. Ogura: Evaluation of advanced pixel-level snakes on cellular hardware platform, in New Circuits and Systems Conference (NEWCAS), 2013 IEEE 11th International, pp. 1–4, 2013.
[7] T. Ikegawa and T. Ogura: CAM$^2$: A highly-parallel two-dimensional cellular automaton architecture, IEEE Trans. Comput., Vol. 47, No. 7, pp. 788–801, 1998.
[8] L. Chua and L. Yang: Cellular neural networks: theory, IEEE Trans. Circuits Syst., Vol. 37, No. 12, pp. 1257–1272, 1988.