EFFICIENT HARDWARE-ORIENTED CELLULAR ACTIVE CONTOURS

David L. Vilariño[*]

Dept. Electronics and Computer Science
University of Santiago de Compostela
E-15782 Santiago de Compostela, Spain
dlv@dec.usc.es

Csaba Rekeczky

Analogical and Neural Computing Laboratory
Computer and Automation Research Institute
H-1111 Budapest, Hungary
rcsaba@sztaki.hu

ABSTRACT

In this paper, a new algorithm for the cellular active contour technique called pixel-level snakes is proposed. The motivation is twofold: On the one hand a higher efficiency and flexibility in the contour evolution towards the boundaries of interest is pursued. On the other hand a higher performance and suitability for its hardware implementation onto a CNN chip-set architecture is required. To illustrate the validity of the proposal some examples and data about the computation time extracted from the implementation of the algorithm on the 64x64 CNNUM chip have been included.

1. INTRODUCTION

An active contour is defined as an elastic curve which deforms controlled by image features and shape constraints to adapt itself to the boundaries of the objects of interest. Among the different approaches the cellular active contours (CAC) appear originally intended to resolve the high computational cost inherent to the classical active contours techniques. They are based on a pixel-level discretization of the contours and on a massively parallel computation on every contour cell which leads to high speed processing without penalizing the efficiency of the contour location.

Up to the present two different cellular active contour approaches have been proposed. In [1] an active wave computing approach is introduced. This consist of a topographic non-iterative region propagation algorithm where the contours come defined by the fronts of travelling waves. This approach has demonstrated a high flexibility in the contour evolution and gives a simple solution to the changes of topology required when two different wavefronts collide. Nevertheless sophisticated stop criteria are usually required to control conveniently the wave-front propagation which may increase considerably the computation complexity in real applications.

In [2] the called pixel-level snakes (PLS) are addressed. They represent a topographic iterative active contour algorithm where the contours are explicitly represented and guided by potential fields. Although this strategy has demonstrated a good performance in multiple applications ([3]), it presents some limitations and exceptions which reduce its efficiency.

Keeping in mind the characteristics of the commented CAC techniques, we propose an improved algorithm for PLS which performs a better contour evolution and a more efficient management of the topological transformations. In addition the hardware implementation of the algorithm is considerably simplified. This has been tested on the 64x64 CNNUM chip ([4, 5]). All the examples used to illustrate the capabilities of the proposal has been obtained from the processing of the algorithm on the chip.

2. IMPROVED PIXEL-LEVEL SNAKES

Into the context of PLS, the active contours are represented as sets of 8-connected activated pixels into a binary image called contour image. This has the same dimensions as the image containing the objects or regions to be defined. The contour evolution consists on an iterative process of activation and deactivation of the contour image pixels along the four cardinal directions. These operations are driven by external information extracted from the image under processing and internal information derived from the contours themselves. The result is equivalent to contour shifts towards final shapes and locations according with the requirements from the guiding information. The guiding information is represented by scalar (or vectorial) potential fields extended to all pixels of the image under processing.

Conceptually the PLS consist of three different modules which interact dynamically: Guiding information, contour evolution and topologic transformations. In the following, the main operations to carry out in these modules are analyzed and their deficiencies are approached.

[*]On stay at the Computer and Automation Research Institute of Hungarian Academy of Sciences in Budapest (SZTAKI), in the framework of the EU Centre of Excellence program.

2.1. Guiding information

The components of the guiding forces along the direction under processing are derived from the external and internal potential matrices by simple directional gradient operations. Since a positive force should indicate a valid direction for the contour evolution only its sign is needed. Therefore in this stage a thresholding operation is also included. These operations are gathered into the called guiding force extraction module (GFE).

The external potential is extracted from the image to be processed and its nature depends on the particular application. It represents an external input to the PLS algorithm. On the other hand the internal potential is derived directly from the active contours. This is estimated by a recursive low-pass filtering or diffusion operation acting on the contour image. The result is a real-valued array characterized by lower potential values into the cavities of the contour and higher outside. Therefore a directional gradient operation acting on this array will originate positive internal forces which push to reduce the local curvature and therefore to smooth the contour shape [3].

Numerous provisions have been made in the literature to improve the robustness and stability of the active contour evolution. Towards this direction in [6] a *balloon force* has been introduced to the parametric deformable models. This new term comes from an anisotropic pressure potential that controls the evolution of the contours helping them to trespass spurious isolated weak image edges and counters their tendency to shrink (due to the internal forces). The PLS can effectively inflate (compress) the contours by adding higher (lower) potential terms to those locations inside the closed curves with respect to those situated outside (Fig. 1). The process is mainly supported by a weighted hole filling operation. The sign of the weight constant will determine the inflating or deflating nature of the potential.

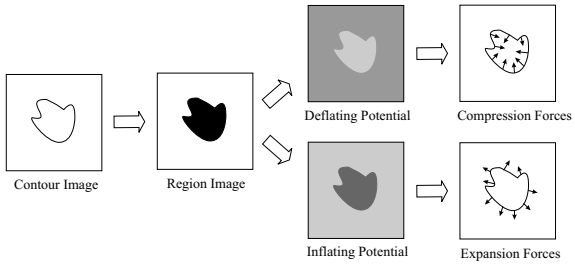

Fig. 1. Generation of inflating/deflating potentials for PLS. Lower potential is represented by higher intensity.

2.2. Contour evolution

To produce the desired contour evolution a directional contour expansion (DCE) of the active contours followed by a directional contour thinning (DCT) are carried out along the direction under processing. In the original PLS algorithm, both DCE and DCT operations can only act on those locations which coincide with activated pixels in the GFE output. Since the expansion and the thinning operations affect to different pixels they rely on different guiding information. Therefore to avoid the appearance of ill-defined contours the DCE operation is restricted to only contour duplications. This approach for the contour evolution provokes the appearance of a *scattering* effect when a contour is anchored in one or more pixels.

We have proved that a non-constrained thinning operation can support the effective contour evolution making the requirement of only contour duplications unnecessary. This small but critically important change leads to a more efficient contour evolution which relies on only one external constraint (the DCE driving). Furthermore, since the contour expansion is not restricted to only duplications the scattering effect disappears as it is illustrated in Fig. 2.

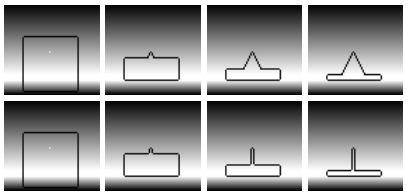

Fig. 2. In this example an active contour evolves towards higher intensities. One pixel in the upside of the contour becomes anchored based on the guiding information which provokes the scattering effect form the original PLS algorithm (first row). This effect does not appear in the improved version (second row).

One important limitation strongly related with the scattering effect, appears in those applications where evolutions along very narrow cavities are involved. The new approach allows to reach deep locations as it is illustrated in Fig. 3.

Not only a higher efficiency in the contour evolution is achieved but also the PLS algorithm is simplified. Now, the DCE module is supported by only one 3x3 directional template per direction instead of the four operations required for the structure in [2].

2.3. Topologic transformations

When the number of active contours does not coincide with the number of objects into the scene the collision between different contours (or different parts of the same contour) may occur. In [2] a solution to manage the changes

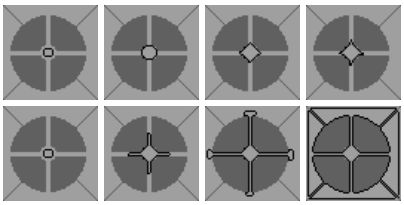

Fig. 3. The contour evolution along very narrow cavities with the original PLS algorithm (first row) is less efficient than that with the new approach (second row).

of topology supported on local CNN-operations was proposed. The approach is based on the preventing of collisions between contours and the controlled split of these contours followed by the merge of the new ones before the collision. This operations of split and merge are supported by the previous detected collision points. Nevertheless not every collision point can guarantee the changes in topology which makes this approach a rather conservative method of resolution of topological transformations (see Fig. 5). On the other hand, the changes of topology are based on *possible* collision points and no on real collisions between contours which have been previously prevented. Therefore they can appear situations where topological transformations are approached even though they are not required (see Fig. 6).

The commented strategy represents a compromise between complexity and accuracy in the management of topological transformations into the contour context where a considerable effort is required to keep well defined the resulting contours. This requirement is dramatically relaxed into a region propagation framework where the contours come defined as frontier pixels of regions into the image space [1]. When two contours collide the collision points do not longer belong to the set of frontier pixels of the associated regions and consequently they neither belong to the set of contour pixels. We propose to face the changes of topology into the region context by means of the three operations indicated in Fig. 4. The contours are transformed in regions by means of a hole filling operation together with a one-step morphological opening (erosion+dilation). Finally they are giving back to the contour context by a binary edge detection which extracts the set of frontier pixels of the regions.

This strategy to manage the topologic transformations has a clearly higher performance than that reported in [2] and attends the changes of topology *whenever* they are required (Fig. 5). On the other hand, since now the handling of the topologic transformation is supported by real collision between contours, the changes of topology are attended *only* when they are actually required (Fig. 6).

The new proposal also outperforms the contour-based approach in the implementation performance: only three isotropic 3x3 linear templates are required instead of the 18

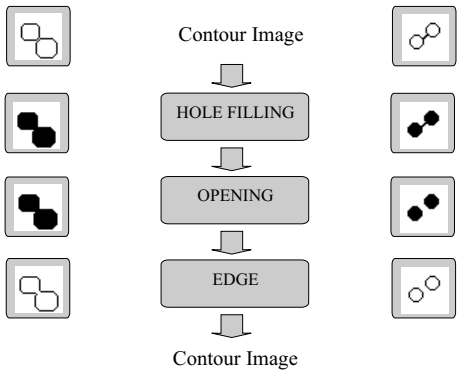

Fig. 4. Flow diagram of the operations for the new topologic transformations module.

directional templates reported in [2].

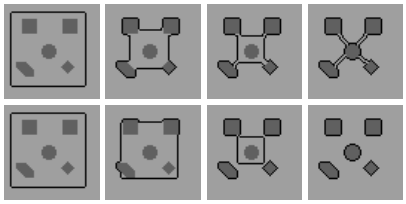

Fig. 5. In this example, the required topological transformations cannot be approached by only local operations based on the proposal in [2] (first row). However the required topological transformations are correctly attended with the new approach (second row).

3. IMPLEMENTATION

Fig. 7 shows the flow diagram of the proposed PLS algorithm PLS intended for its implementation onto the CNNUM [4]. For the CNN operations the initial state, the input and the fixed state map are labelled with A, B and F respectively. The result of those CNN operations where the A or B labels are missing is independent of the initial state or the input. The dark gray items represent the external data provided by the user. They include input images (the external potential and the initial contour images), weights (k_{ext}, k_{int} and k_{inf} to weigh the influence of the external potential, internal potential and inflating/deflating potential respectively) and the switch s_{inf} to select between inflating (+1) and deflating (-1) potential, estimated in the balloon potential estimation module (*BPE*).

The templates for the DCE and DCT operations are derived by the rules showing in Fig. 8. The directional gradient (*D_Gr*) is performed by the Sobel operator. The remainder of the processing steps consists of simple binary logical

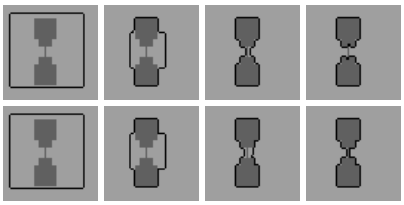

Fig. 6. With the original PLS algorithm a change of topology appears even though it is not required (first row). With the new proposal only changes of topology are attended where real contour collisions appear (second row).

operations and well-known propagative and non-propagative analogic CNN operations. In Table 1, the execution times for each module of the algorithm in Fig. 7 are gathered.

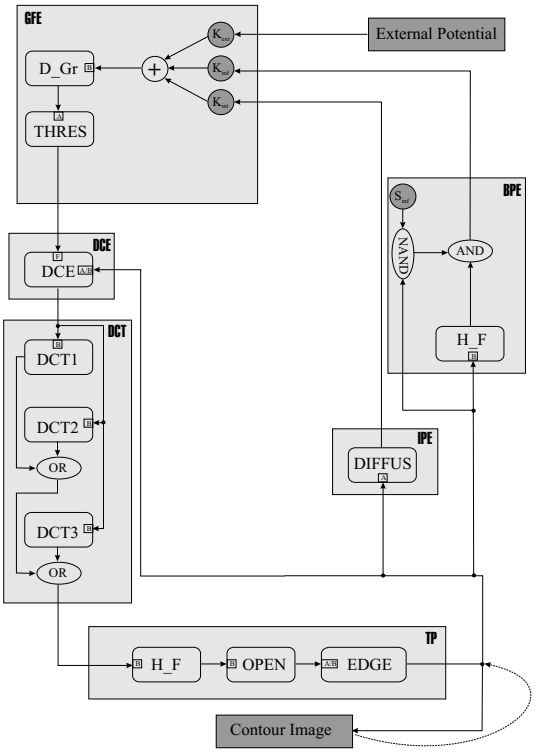

Fig. 7. Flow diagram of the new PLS algorithm containing all the implemented CNN operations.

4. CONCLUSIONS

Pixel-level snakes represent a cellular active contour technique which has demonstrated a high performance in multiple active contour applications. In this paper, the associated algorithm has been analyzed and some limitations and exceptions have been discussed leading to new proposals to increase the performance of the PLS. The new algorithm with all the functionalities requires less than $4ms$ to complete one cycle running in the 64x64 CNNUM chip [5]. We have observed that in real time applications like video object segmentation and tracking less than ten iterations per frame are usually needed. Therefore, even with the full version of the algorithm the processing of 25 frame/s is easily achieved. Furthermore in the most of the practical cases not all the functionalities are needed at the same time which considerably reduces the computational effort.

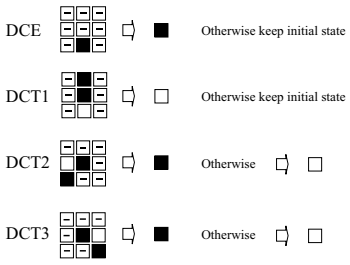

Fig. 8. Patterns for DCE and DCT (North direction).

IPE	$40\mu s$	**DCE**	$60\mu s$
BPE	$750\mu s$	**DCT**	$160\mu s$
GFE	$250\mu s$	**TP**	$760\mu s$

Table 1. Time of processing of the modules in the PLS algorithm extracted from the implementation on the 64x64 CNNUM chip (one iteration along a cardinal direction).

5. REFERENCES

[1] C. Rekeczky and L.O. Chua, "Computing with Front Propagation: Active Contour and Skeleton Models in Continuous-Time CNN," *Journal VLSI Signal Proc. Syst.*, vol. 23, no. 2/3, pp. 373–402, 1999.

[2] T. Kozek and D.L. Vilariño, "An Active Contour Algorithm for Continuous-Time Cellular Neural Networks," *Journal VLSI Signal Proc. Syst.*, vol. 23, no. 2/3, pp. 403–414, 1999.

[3] D.L. Vilariño, D. Cabello, X.M. Pardo and V.M. 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] T. Roska and L.O. Chua "The CNN Universal Machine: An Analogic Array Computer", IEEE Trans. Circ. Syst. II, vol. 40, no. 3, pp. 163–173, 1993.

[5] S. Espejo et al. "A 64x64 CNN Universal Chip with Analog and Digital I/O," *ICECS98*, pp. 203–206, 1998.

[6] L.D. Cohen and I. Cohen, "Finite Element Methods for Active Contour Models and Ballons for 2D and 3D Images," *IEEE Trans. PAMI*, vol. 15, pp. 1131–1147, 1993.