Evolution of Pixel Level Snakes towards an efficient hardware implementation

David Lopez Vilariño
Department of Electronics and
Computer Science
University of Santiago de Compostela
Santiago de Compostela, Spain
Email: dlv@dec.usc.es

Piotr Dudek
School of Electrical and
Electronic Engineering
University of Manchester
Manchester, United Kingdom
Email: p.dudek@manchester.ac.uk

***Abstract*— Since Pixel Level Snakes (PLS) were introduced, several algorithms implementing this cellular active contour technique have been proposed. In this paper, we review the main features of these algorithms and propose some modifications to optimize the computation performance of PLS when they are executed on fine-grain pixel-parallel processor arrays. The modified algorithm has been implemented on a focal plane cellular processor array (SCAMP-3 vision chip) and tested on several applications of practical interest.**

I. Introduction

Pixel-level snakes (PLS) are an active contour technique inspired by the energy-based deformable models. PLS are based on three conceptual modules: guiding information extraction, contour evolution and topological transformations (Fig. 1). The input consists of a binary image containing the active contours and a multibit image containing information from the image under processing, along with some parameters which configure the control of the contour evolution. The implementation consists of an iterative algorithm which operates along the four cardinal directions. The contour evolution is performed as a pixel-by-pixel shift (activation and deactivation of pixels of the binary contour image) towards the features of interest, (*e.g.,* boundaries of objects). The binary image given back as output after each cycle (four iterations, one for each cardinal direction) contains the active contours slightly shifted and/or deformed to fit the objective regions. Therefore the final result is obtained after several cycles.

Since the PLS technique was introduced, the associated algorithm has undergone several modifications in order to improve its performance from the point of view of either the image processing or hardware implementation.

The seminal fully-operational approach (PLSv1) is addressed in [1]. PLSv1 retains the structure of Fig.1. The guiding information extraction module includes two sub-modules:

- Internal Potential Estimation (IPE)
- Guiding Force Extraction (GFE)

In the PLS approach, active contours evolve due to the influence of forces derived from a potential field that is projected onto the contour image. This potential field results from the combination of the so called external potential field (derived from the image under processing) and the internal potential field estimated from the contours themselves. The external forces push the active contours towards the features of interest. The internal forces keep the contour shape smooth.

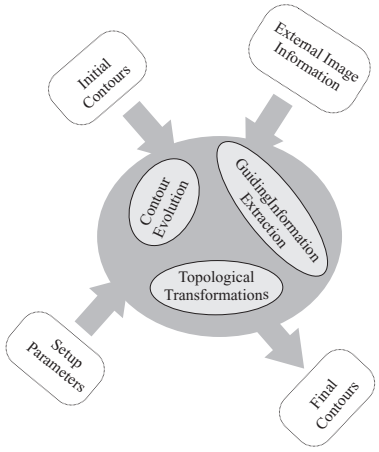

Fig. 1. Pixel-level snakes conceptually consist of three modules interacting dynamically along the four cardinal directions.

The contour evolution consists of two sub-modules:

- Directional Contour Dilation (DCD): the contours are dilated in the direction of processing according to the guiding information.
- Directional Contour Thinning (DCT): The contours are thinned along the direction of processing according to the guiding information

The combination of both operations leads to the activation and deactivation of pixels of the binary contour image and as a consequence the contour shift and/or deformation based on the guiding forces.

Finally, the topological transformation module includes two sub-modules:

- Collision Point Detection (CPD): Collision between contours are avoided through the creation of an infinite potential barrier (one pixel wide) between contour sections.
- Collision Resolution (CR). Controlled topological transformations are carried out by split and merge operations.

Further details about PLSv1 can be found in [1].

1-4244-0921-7/07 $25.00 © 2007 IEEE.

PLSv1 has demonstrated a good performance in some applications where active contours are frequently used. Nevertheless, it has some drawbacks and exceptions which reduce the efficiency of the algorithm. In order to remove some of these limitations a new version of the algorithm (PLSv2a) was proposed in [2]. PLSv2a retains the structure of PLSv1 (based on three main modules) with the following contributions:

- New term to guide the contour evolution (balloon potential). The expansion or compression effect of the contour shape can be achieved by this potential field.
- DCT independent on the guiding information. It provides more flexibility in the contour evolution.
- New strategy to tackle the changes in the contour topology. A new module (TP) takes the active contours as input and redefines those as frontier pixels of the corresponding enclosed areas.

The approach of topological transformations in PLSv2a makes the CPD module unnecessary . However it was kept to allow more capabilities of the algorithm (making possible the preservation of the topology when required). A comprehensive comparison between PLSv1 and PLSv2a can be found in [2].

In the PLSv2a algorithm, the use of regions surrounded by the active contours simplifies the execution of the topologic transformations as well as the estimation of the balloon potential. Such regions are obtained by using hole filling operations. In theory a CNN-type processor array can compute the hole filling operation in a single step. However, current CNN-based chips implement that operation in an iterative way, making it very expensive in computational effort. Furthermore, the hole filling operation introduces an important constraint in the image processing: it is not possible to manage active contours evolving inside other contours.

In this paper we propose an approach which overcomes these constraints. This variation of the PLSv2 algorithm (let us name it PLSv2b) allows to reduce the computational effort and improves the active contour performance. We propose to replace the hole filling operation by some non-propagative operations together with a local logic memory (LLM) to update the regions associated with the contours after each iteration instead of computing them only from contour images. Fig. 2 shows the flow diagram of the modified algorithm and Fig. 3 the module responsible of topological transformations. In a given iteration, the image containing the regions surrounded by the active contours of the previous iteration are stored in LLM. Then, after the directional contour dilation new pixels are added to these regions stored in LLM. After the directional contour thinning some pixels have to be removed from the regions. To do that, first the image stored in LLM is eroded and then the result is added to the DCT output. Finally, this binary image which represents the region image of the current iteration is stored in LLM. From this image we can estimate the balloon potential and update the active contours (by means of a binary edge detection) for the next iteration.

Note that, as a difference with previous PLS algorithms, a region image is introduced as input instead of an initial contour image.

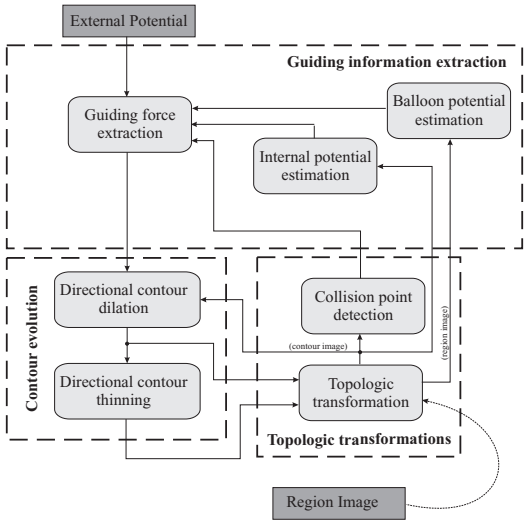

Fig. 2. Flow diagram of the PLSv2b algorithm.

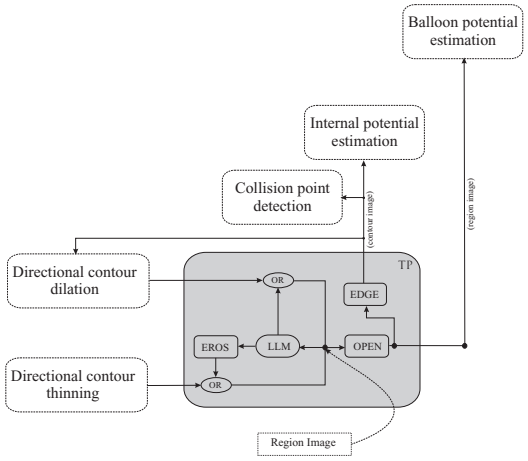

Fig. 3. New module for handling the changes in the contour topology. *EROS* and *OPEN* represent morphological erosion and opening operations respectively.

Note that the extra local logic memory required in PLSv2b does not represent a limitation for the implementation of the algorithm in vision systems like SCAMP-3 [4] where the number of memories is enough to run the complete algorithm.

Another approach, which also overcomes the aforementioned limitations of PLSv2a, was recently proposed in [5]. In that version (PLSv3), the active contours are implicitly defined as the boundaries of active regions. The PLSv3 approach eliminates the need for hole-filling operation and simplifies the contour evolution step (eliminating contour thinning operations) since the one-pixel contour width and its continuity are implicitly guaranteed. However, the overall computational cost is not necessarily lower, because a global cycle along the four cardinal directions requires eight steps of the directional dilation, against only four of the previously commented PLS algorithms.

In the next section a comparison between the various PLS algorithms is discussed. A discussion about the performance of PLS against conventional active contours techniques can be found in [2].

II. Comparison

We have considered three implementations of PLS (Table I). The first approach, PLSv1, is completely based on contour evolution. The second proposal, PLSv2 (in both variants PLSv2a and PLSv2b), represents a hybrid approach managing together region and contour information. Finally the third proposal, PLSv3, is a fully region-based approach (except for the internal potential estimation).

TABLE I
THREE IMPLEMENTATIONS FOR PLS

Implementation	Model Evolution	Topological Transformation
PLSv1	contour-based	contour-based
PLSv2	contour-based	region-based
PLSv3	region-based	region-based

As it is discussed in [2], PLSv2a clearly improves upon PLSv1 except,

- in those applications needing to manage contours completely surrounded by other active contours, and
- in those applications needing to manage open contours.

PLSv2b and PLSv3 overcomes the first of these limitations (Fig. 4). In fact, this represents the main contribution of these algorithms from the point of view of the image processing.

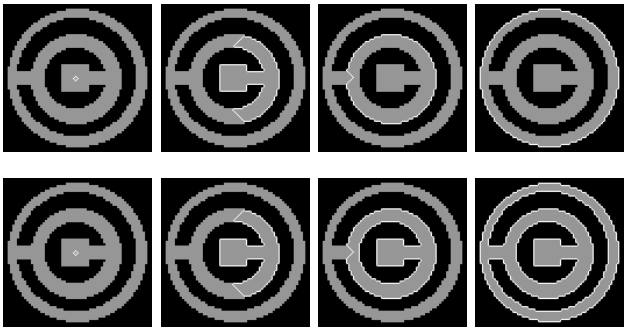

Fig. 4. Contour evolution. Hole Filling removes internal active contours in PLSv2a computation (first row). It does not happen with PLSv2b or PLSv3 (second row).

The computation of open contours is also a limitation in PLSv3. This algorithm requires two sets of four iterations to complete a cycle. Between these sets the regions enclosed by the active contours are inverted and, depending on the inversion strategy used, it leads to either a complete removal of open contours or a possibility of open contours bifurcating and fragmenting.

On the other hand, PLSv2 algorithms can compute open-contours easily because of the explicit definition of the active contours in the evolution stage and the explicit constrains of the contour connectivity. However, these algorithms cannot manage the possible topological transformations. In both algorithms, the module responsible of handling changes in topology (TP) is intended to operate on regions. This module includes a morphological opening operation and a binary edge detection which will return wrong outputs if they process contours instead of regions.

Although PLSv2a and PLSv2b only overcome partially the limitation of the management of open contours, it is sufficient for some applications such as robot path planning where the optimal route between two points (start and target) is required. Given a floor plan the objective is to join a start and a target points along the shortest route. This complex task can be approached with the PLSv2 algorithms based on three computing steps with different configurations of the externally accessible parameters (Fig. 5):

1) Contour evolution based on inflating potential . The collisions should be avoided (CPD ON).
2) Contour evolution based on deflating potential. The collisions should be avoided (CPD ON).
3) Contour evolution based on internal potential. The collisions should be allowed (CPD OFF).

In the three steps, the labyrinth shape (introduced as external potential) constrains the contour evolution in such a way that the contours are forced to flow inside the labyrinth like wavefronts. Details of this application can be found in [3].

With PLSv3, the first two processing steps for the estimation of a quasi-optimal route can be implemented. Yet, the third step requires the use of open-contours that cannot be managed by this PLS version. The contribution of the third step is particularly clear in sparse labyrinths (Fig. 6).

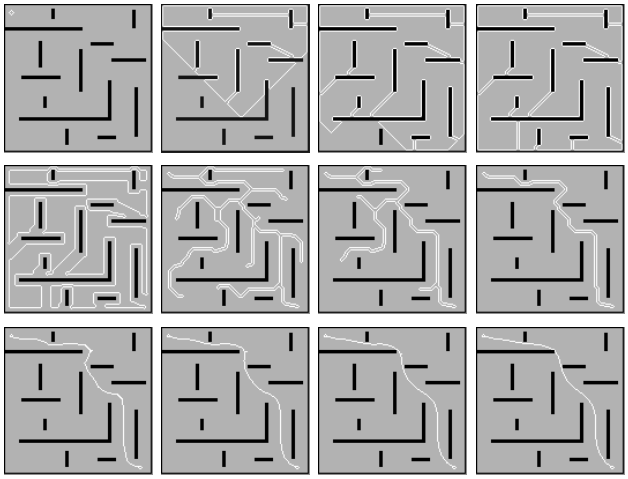

Fig. 5. Shortest path in a labyrinth computed with PLSv2b.

Finally all PLS algorithms have the same size limit of three pixels for the regions enclosed by the active contours (including contours themselves) which prevents them from being able to propagate into cavities with less than three pixels

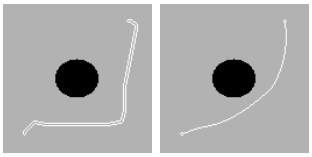

Fig. 6. Shortest path in a sparse labyrinth computed with PLSv2b. Picture on the left shows the output of the second step and that on the right the output of the third step.

of width. In case of contour-based PLS it is a consequence of the requirement of well-defined closed contours [2], [3]. On the other hand PLSv3 cannot operate correctly in such scenes because of the active region inversion operation which results in the elimination of regions with one or two pixels of width (although they can operate in case of unidirectional contour evolution, *i.e.*, expansion only which enables 1-pixel wide contours [5]).

As a summary we can conclude that the PLSv2b algorithm is the most versatile version of PLS. However, it does not mean that it is the better choice for every application.

We have implemented the PLSv2b and PLSv3 algorithms on an SIMD processor array vision chip, SCAMP-3 [4]. The chip executes a sequence of simple array instructions (addition, inversion, one-neighbour access), which operate in a pixel-parallel fashion on a 128x128 array, at a rate of 1.25 MOPS per pixel. The time required to implement the main components of the PLSv2b algorithm are given in Table II. The overall contour evolution takes $518.4\mu s$ per complete cycle (evolution in four directions). This means that many iterations of evolution can be executed within a single video frame, enabling the use of PLS technique in real-time computer vision applications.

TABLE II

EXECUTION TIMES FOR THE ALGORITHM PLSV2B IMPLEMENTED ON THE SCAMP-3 VISION SYSTEM.

MODULE	ITERATION	CYCLE
GFE	6.4 μs	25.6 μs
DCD	6.4 μs	25.6 μs
DCT	27.2 μs	108.8 μs
CPD	16 μs	64 μs
TP	40 μs	160 μs
BPE	-	4 μs
IPE	-	130.4 μs
Total		518.4 μs

As compared with PLSv3, the PLSv2a algorithm exhibits similar performance. In practice, the execution times for a particular implementation will depend on the processor architecture, as well as the chosen strategy for calculating collision detections, topology transformations, internal potentials, etc. However, in many cases the PLSv2b can outperform PLSv3. For example, the PLSv3 algorithm implemented in [5] requires $546\mu s$ for a complete cycle, as compared with $518\mu s$ for PLSv2b. More importantly, the PLSv2b is more versatile, enabling the processing of open contours.

In order to illustrate the PLS computation of the SCAMP-3 system, figure 7 shows some frames of an experiment where some objects are segmented and tracked with PLS, executing 30 iterations per frame at 25 frames per second.

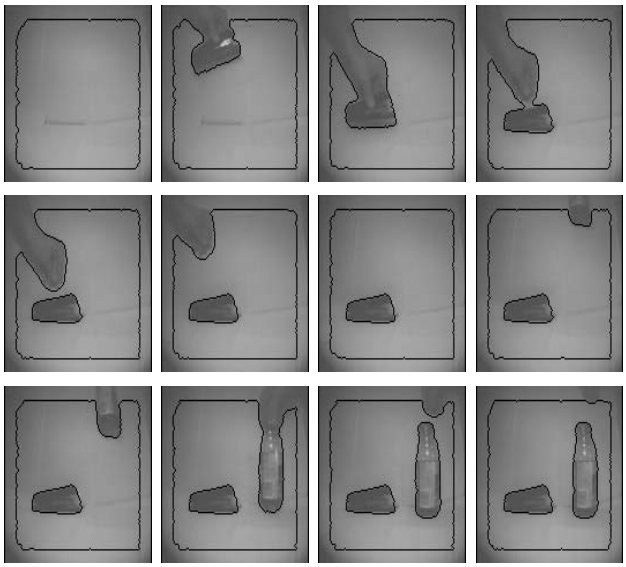

Fig. 7. Several snapshots resulting from the real time processing of PLS on the SCAMP-3 vision system.

III. CONCLUSIONS

Since PLS were introduced, the associated algorithm has undergone some modifications to improve the capabilities as active contour technique and/or to simplify the hardware implementation. In this paper some proposals to implement PLS are discussed. We propose a new version of the algorithm, PLSv2b, which is an adaptation of the previous version PLSV2a where the hole-filling operation is replaced by local non propagative operations in order to improve the performance of the implementation. The algorithm exhibits improved performance and is particularly suitable for real-time image processing on cellular processor arrays.

REFERENCES

[1] 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, N2, pp.189–204, 2003.

[2] D.L. Vilariño and Cs. Rekeczky: "Pixel-Level Snakes on the CNNUM: Algorithm Design, On-Chip Implementation and Applications", International Journal of Circuit Theory and Applications, Vol.21, N.2, pp.189-204, 2005.

[3] D.L. Vilariño and Cs. Rekeczky. Shortest Path Problem with Pixel-Level Snakes: Application to Robot Path Planning. In *Int. Workshop on Cellular Neural Networks and their Applications, CNNA2004*, pp. 135–140, 2004.

[4] P.Dudek and S.J. Carey: "A General Purpose 128x128 SIMD Processor Array with Integrated Image Sensor ", Electronics Letters, Vol.42, N.12, pp.678-679, 2006.

[5] P.Dudek and D.L. Vilariño: "A Cellular Active Contours Algorithm Based on Region Evolution", IEEE Int. Workshop on Cellular Neural Networks and Their Applications, CNNA2006, pp.269–274, 2006.