2006 10th International Workshop on Cellular Neural Networks and Their Applications, Istanbul, Turkey, 28-30 August 2006

A Cellular Active Contours Algorithm Based on Region Evolution

Piotr Dudek*, David Lopez Vilariño+
*School of Electrical and Electronic Engineering, University of Manchester, United Kingdom
e-mail: p.dudek@ manchester.ac.uk
+Department of Electronics and Computer Science, University of Santiago de Compostela, Spain
e-mail: dlv@dec.usc.es

***Abstract*—This paper introduces a new algorithm for implementing cellular active contour technique based on pixel-level snakes (PLS). The main motivation is the optimization of the computational performance, especially when PLS are implemented on pixel-parallel single instruction multiple data (SIMD) processor arrays. The algorithm is based on the evolution of an active region. This allows the implementation of the entire algorithm using very simple local rules. Additionally, nested contours and propagation into narrow cavities are supported. The algorithm and experimental results from real-time implementation on vision chips are presented.**

I. INTRODUCTION

Active contours [1] are elastic curves which deform controlled by image features and shape constraints to adapt themselves to the boundaries of the regions of interest. The deformable contour techniques are usually classified as either energy-based [2] or level-set based [3]. Both types differ in the contour representation, implementation and capabilities from the image processing point of view. However, they share high computational requirements which make them unsuitable for applications requiring short response time.

Cellular active contours (CAC) were introduced to reduce the computational effort inherent in the conventional active contour techniques. They are based on a pixel-level discretization of the contours and a massively parallel computation. Processing every contour cell concurrently leads to a higher processing speed without penalizing the efficiency of the contour evolution. Currently there are two different CAC approaches. First, a PDE-based approach proposed in [4] which implements active contours via a non-iterative region propagation technique, where the contours are defined as the fronts of the propagating trigger-waves. Like the classic implicit active contour techniques, this approach gives a very simple solution to the possible topologic transformations, but, in contract to those techniques, the speed of computation is very high. The main weakness of this approach comes from the difficulty in the control of the contour evolution. It does not allow the simultaneous expansion and contraction of different parts of the active regions. Furthermore, usually it requires sophisticated stop criteria to control the active wave propagation which can lead to a significant increase of the computational complexity in real applications.

On the other hand, the so called pixel level snakes (PLS) [5] are an iterative CAC technique where the contours are explicitly represented and evolve guided by local information and regularizing terms dependent on the contour curvature. The iterative computation of PLS allows an efficient control of the contour evolution. However, as a consequence of the explicit representation of the active contours, the algorithm complexity of PLS is clearly higher than the active wave based approaches [6].

In this paper, a CAC technique situated mid-way between the above described techniques is proposed. In a way similar to the active wave computing technique, the active contours are represented as the boundaries of active regions, thus reducing complexity of the contour evolution. On the other hand, keeping the iterative computation of PLS a versatile and simple control of the contour deformation is achieved

II. PIXEL LEVEL SNAKES

The proposed algorithm builds upon the overall method of PLS introduced in [5]. In a PLS algorithm the contour is represented explicitly by pixels in an array corresponding topographically to the input image and the contour evolution is implemented by one-pixel shifts of the contour in this array. The overall flow of a PLS algorithm is presented in Figure 1. It is an iterative algorithm, in which the contour pixels evolve according to external and internal forces. The potential field is a topographic map, calculated as a weighted sum of External Potential, Internal Potential and Balloon Potential. This potential field produces forces that guide the contour evolution towards the minimum energy level.

The External Potential is calculated from the input image, and determines the overall target for the contour evolution, to fulfill the required segmentation task determined by the particular application. For example, the contour may track the boundaries of all objects, the edges of selected objects in the image (for example moving objects, or one marked object only), etc. Standard low-level image processing is performed on the input image to determine the External Potential (the operations may include noise filtering, edge-detectors, distance transforms, etc.). In some applications, involving static images, External Potential is calculated once. In other applications, involving moving images (e.g. real-time computer vision) the External Potential is calculated for each image frame. The External Potential can be updated on each iteration of the PLS evolution, or several iterations of PLS evolution can be performed for each External Potential.

1-4244-0640-4/06/$20.00 ©2006 IEEE

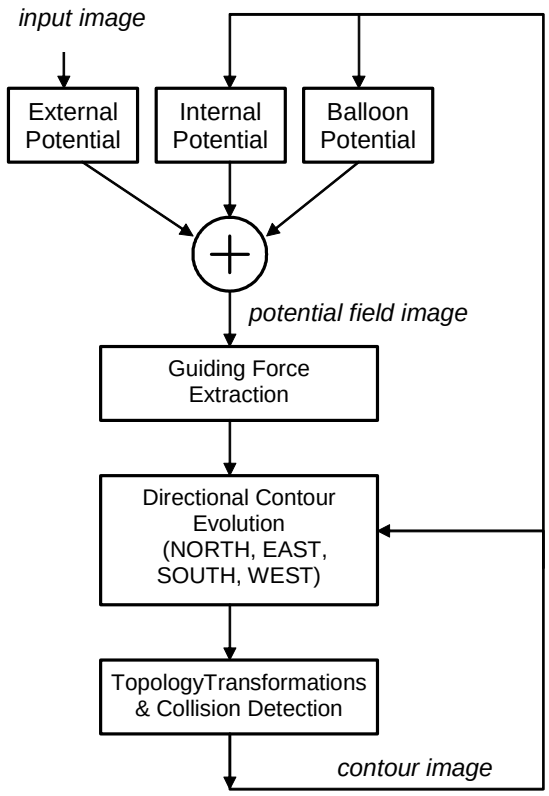

Fig.1. Overview of a pixel-level snakes (PLS) algorithm

The Internal Potential is a topographic map obtained by diffusing the contour image. The resulting potential field helps to keep a smooth shape of the contour.

The Balloon Potential produces additional inflating/deflating forces that are sometimes needed to guide the contour evolution towards the target, especially if the contours are initialised far from the desired minima of the External Potential field and could settle at some local minima locations, or if the gradients of the External Potential field are zero in some image areas.

The relative weights of External, Internal and Balloon Potential are adjusted, depending on the application. (In some situations, some of the weights can be zero, which obviously means that the corresponding potential does not have to be calculated, thus simplifying the overall algorithm)

Contour evolution is the main component of a PLS algorithm. It operates along the four cardinal directions in sequence. On each iteration, contour evolution is performed as a pixel-by-pixel shift of the contour image, towards a lower potential location. (Internal Potential and Balloon Potential can be updated on each iteration, or once for every four cardinal directions, to reduce computations). The requirement of maintaining one-pixel wide contour is the main difficulty when implementing a PLS algorithm, and the methods proposed earlier were based on a number of local rules to handle Directional Contour Expansion and Directional Contour Thinning. Furthermore, merging and splitting of contours had to be handled by a separate Topologic Transformations module, which could contain Collision Point Detection module (used in applications that require avoiding collisions between contours) and algorithms to handle splitting and merging of contours.

III. Proposed Algorithm

The algorithm proposed in this paper follows the general flow of the PLS algorithm, but is based on the evolution of an active region. The contour is defined implicitly as the boundary of the region. This greatly simplifies topographic transformations (merging and splitting of contours), as well as implementation of inflating/deflating forces. The main advantage of this approach is the elimination of the hole-filling step in the original PLS algorithm [5], which is time-consuming when performed iteratively on a processor array. It should be noted, that in theory a CNN-type array processor could handle hole-filling in a single operation, using the 'hole_fill' template. However, the difficulty in implementing an 'ideal' CNN processor in silicon [7,8] leads to the situation that hole-filling still has to be implemented iteratively, even on the most recent CNN chips [6]. Simple, robust solutions for hole-filling exist, based on asynchronous cellular logic [9]. Nevertheless, an approach which eliminates hole-filling leads to an algorithm suitable for many present-day processor arrays, such as [10-13], as well as implementation of pixel-level snakes in software, on conventional microprocessors.

Furthermore, the proposed algorithm implements contour evolution using very simple local rules, which results in a fast implementation. In contrast to previously reported implementations, the proposed algorithm also enables nested contours (closed contours within other contours), and enables contour propagation inside 1-pixel wide cavities.

A. Contour Definition

Instead of evolving 1-pixel wide contours, the method presented in this paper relies on the evolution of active regions. The overall topographic map is initially segmented into two sets of pixels, one set are pixels belonging to the 'active regions', the remaining pixels are the 'background regions'. We can represent the regions using a binary array **a**, where the active regions are '1', and the background regions are '0'.

The contour could be defined implicitly, as the boundary between two regions, as illustrated in Figure 2a. To maintain the one-pixel wide contour (useful for display purposes, and producing output compatible with previous PLS algorithms, and also necessary to calculate Internal Potential) we assume that the 1-pixel wide contour is defined as a boundary of the 'active region', and lies within the active region, as illustrated in Figure 2b. In general, there may be multiple disjoint active regions in an image, resulting in multiple closed contours, as shown in Figure 3a. This definition supports also a situation, where a contour in placed inside another contour, as shown in Figure 3b.