A Fast Snake Algorithm for Tracking Multiple Objects

Hua Fang*, JeongWoo Kim* and JongWhan Jang*

Abstract—A Snake is an active contour for representing object contours. Traditional snake algorithms are often used to represent the contour of a single object. However, if there is more than one object in the image, the snake model must be adaptive to determine the corresponding contour of each object. Also, the previous initialized snake contours risk getting the wrong results when tracking multiple objects in successive frames due to the weak topology changes. To overcome this problem, in this paper, we present a new snake method for efficiently tracking contours of multiple objects. Our proposed algorithm can provide a straightforward approach for snake contour rapid splitting and connection, which usually cannot be gracefully handled by traditional snakes. Experimental results of various test sequence images with multiple objects have shown good performance, which proves that the proposed method is both effective and accurate.

Keywords—Snake, Detection, Tracking, Multiple Objects, Topology Changes

1. INTRODUCTION

The snake algorithm introduced by Kass et al. [1] has been widely applied in various object contour detection and object tracking. Xu and Prince [2] introduced gradient vector flow (GVF) as a new external force in the snake energy. The method has the advantages of insensitivity to initialization as well as the ability to move into boundary concavities, but it cannot handle the gourd-shaped concavities. Kim et al. [3] presented an improved snake-based method to handle the problem of highly gourd-shaped concavities using the Frenet formula.

In recent years, the snake algorithm has been developed for tracking object contours [4-5]. The resulting snake of frame $t-1$ is used as an initialization for the snake of frame t. The behavior of a snake in tracking an object will also depend on the movement of the object. In the original works, the snake is unable to track a contour if the initialization provided by the contour detected in the previous frame is not close enough to the newly displaced object's boundary in the current frame. GVF has increased the probability for the convergence of the snake to the object's boundary, even if the snake is not initialized close to the boundary. However, if there is more than one object in the image and the snake is partly located between two objects, the GVF force field pulls the snake to both objects to result in a wrong detection. [6] improved a new force to block strong deformations of the snake contour and the snake is moved as a rigid body towards the object to be tracked. However, if one of the objects moves very close to intruding into another's re-initialized contour, this method has no longer an effect.

Manuscript received March 7, 2011; first revision April 14, 2011; accepted May 9, 2011.
Corresponding Author: JongWhan Jang
* Dept. of Information and Communications Engineering, PaiChai University, Daejeon, Korea (fanghua @pcu.ac.kr, wo0905@hanmail.net, jangjw@pcu.ac.kr)

A major shortcoming of traditional parametric snakes is their inability to deal with topology changes. Some research solutions have been proposed to handle this problem such as geometric/geodesic active contour (GAC) [7-8]. The GAC is derived from the classical energy based snake:

$$E(C) = \alpha \int_0^1 |C'(q)|^2 \, dq + \beta \int_0^1 |C''(q)|^2 \, dq - \lambda \int_0^1 |\nabla I(C(q))| dq \tag{1}$$

by considering $\beta = 0$. The GAC snakes are typically implemented using level set techniques where the snake is embedded as the zero level set of a higher dimensional function. Its' zero level curve moves in the normal direction of gradient and therefore stops on the desired boundary. This implementation enables GAC snakes to handle boundary concavities and topology changes naturally [9-10]. On the other hand, during the evolution of calculating the level set function, it is required to update the function value of the level set for all pixels in the whole image. The high cost of computation remains a challenge, which cannot be treated through modifications to the GAC model.

In contrast with GAC, the parametric snakes only need to calculate the energy functions of 8 neighbors. With the advantage of less cost, this approach can be better applied to tracking objects. In this paper, to overcome the limitation of topology changes, a parametric snake-based algorithm has been proposed for automatically tracking multiple objects. Also, the proposed algorithm is different from the "multiple snakes" methods [11]. "Multiple snakes" always requires manually drawing initial contours for each object of interest and lacks the ability to handle the topology changes. We are presenting an automatic snake model, which can extract multiple objects in the region of interest of an image without any manual assistance. If a single snake contour encloses multiple objects of interest, it will split into two smaller snakes. Each new contour repeats the splitting and connecting process until all of the objects are detected. If two snake contours intersect, these two snake contours break at the intersection and merge into one. Compared with the traditional snakes, this proposed method is able to handle the snake topological changes rapidly and automatically. Experimental comparison proves our method is faster and more efficient. Moreover, the proposed contour evolution is topology independent. That is, there is no need to know the prior topology to get the solution. This allows it to detect any number of objects in the image, without knowing the exact number.

This paper is organized as follows: in Section Two, we introduce a new energy term of movement direction for the snake energy function. In Section Three, we propose a fast splitting and connecting snake algorithm depending on the intersection of two segments. In Section Four, we present the performance evaluation of our method and we discuss our conclusions in Section Five.

2. ACTIVE CONTOUR MODEL WITH DIRECTIONAL ENERGY

The active contour model, more widely known as the snake model, in the discrete formulation, the contour is represented as a set of snake points $v_i(x_i, y_i)$ for $i = 0,\ldots,N-1$ where x_i and y_i are the x and y coordinates of the i^{th} snake point, respectively and N is the total number of snake points. The energy function, which minimize the snake is expressed as follows:

$$E_{snake}(v) = \sum_{i=0}^{N-1} (E_{internal}(v_i) + E_{external}(v_i)) \tag{2}$$

where, $E_{internal}$ represents the internal energy term and $E_{external}$ represents the external energy term. The internal energy is composed of two terms.

$$E_{internal}(v_i) = E_{continuity}(v_i) + E_{curvature}(v_i) \tag{3}$$

The main role of the conventional continuity energy is to make even spacing between the snake points by minimizing the difference between the average distance and the distance between neighboring snake points. The main role of the curvature energy term is to form a smooth contour between neighboring snake points.

2.1 Estimation of the Optical Flow

Given the gradient at a pixel location and the pixel value change from one to another, the direction and the magnitude of pixel motion is estimated for individual pixels [12]. The optical flow methods try to calculate the motion between two image frames, which are taken at times t and $t + \delta t$. For a $2D + t$ dimensional case a pixel at location (x, y, t) with intensity $I(x, y, t)$ will have moved by $\delta x, \delta y$ and δt between the two frames, and the following image constraint equation with the Taylor series can be given:

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + \varepsilon \tag{4}$$

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0 \quad \text{or} \quad \frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t}\frac{\delta t}{\delta t} = 0 \tag{5}$$

which results in:

$$\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} = 0 \tag{6}$$

where v_x, v_y are the X and Y components of the velocity of $I(x, y, t)$ and $\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the derivatives of the image at (x, y, t) in the corresponding directions. The velocity $V(v_x, v_y)$ is estimated from:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \sum_{x,y \in R} I_x^2 & \sum_{x,y \in R} I_x I_y \\ \sum_{x,y \in R} I_y I_x & \sum_{x,y \in R} I_y^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_{x,y \in R} I_x I_t \\ -\sum_{x,y \in R} I_y I_t \end{bmatrix} \tag{7}$$

2.2 Improved Snake Energy

To apply the snake to object tracking, the idea is to use the resulting contour of a frame as ini-

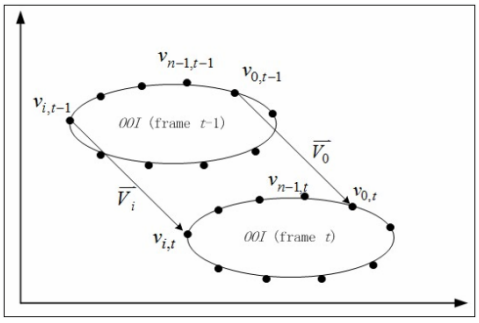


Fig. 1. Object motion and its velocity vector

tialization for the next. Therefore, the result of tracking is depending on the speed of convergence of the snake in every frame.

First, we note that the definition of the snake point in the 2-D Euclidean space at frame t becomes $v_{i,t} = (x_{i,t}, y_{i,t})$. The movement of objects is illustrated in Figure 1, which shows an OOI (object of interest) that has moved from its location in frame $t-1$ to a new location in frame t. Our purpose is to find an estimate of the new location of the snake points based on the velocity generated by optical flow. Suppose that $\vec{V}_i$ is the velocity vector of the snake point v_i from $t-1$ to t. $\vec{u}_i$ is the unit vector of $\vec{V}_i$. The *directional energy* of the snake point v_i, which is going to be minimized, is defined as:

$$E_{direction}(v_i) = -(\overrightarrow{v_{i,j}v_{i,j+1}} \bullet \vec{u}_i) \tag{8}$$

where $\bullet$ is inner product, $v_{i,j}$ means i^{th} snake point and j^{th} iteration. $\overrightarrow{v_{i,j}v_{i,j+1}}$ is the vector from $v_{i,j}$ to $v_{i,j+1}$. The complete snake energy can now be given as:

$$E_{snake}(v) = \sum_{i=0}^{N-1} (E_{internal}(v_i) + E_{external}(v_i) + E_{direction}(v_i)) \tag{9}$$

3. An Approach of Splitting and Connecting Contours

Traditional snake algorithms often lack the capability to process multiple objects. An automatic snake model that can perform a flexible topology changes without any manual assistance is now presented.

Figure 2 (a) shows there are two objects (1 and 2) enclosed in one snake contour. The snake points v_0, v_1,…, v_{N-1} are the initial snake points, and the closed contour linking these points defines an initial snake contour C_0. Line segment S_i connects v_i and v_{i+1}. The segment vector $\overrightarrow{S_i}$ points from v_i to v_{i+1}. The snake contour converges to the object's boundaries by minimizing the energy terms. As shown in Figure 2 (b), the segments of the snake contour intersect at the gap region. Segment $v_{k-1}v_k$ touches segment v_iv_{i+1} at point v_k at the j^{th} iteration. (The iteration of snake points is calculated by a 3×3 window.) After the splitting and connecting process, the initial snake contour is divided into two new smaller contours to determine the corresponding contour of each object.

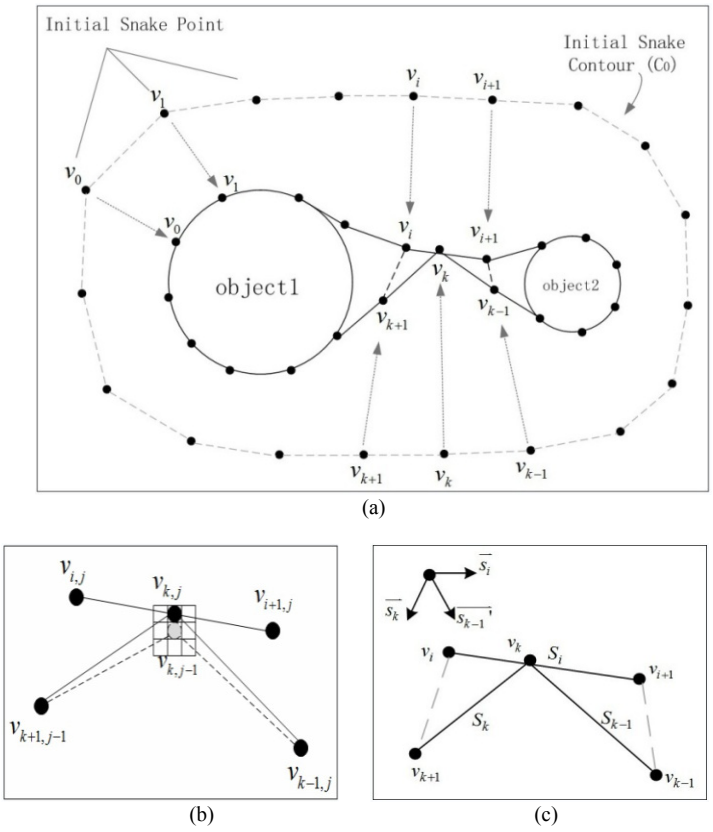


Fig. 2. (a) The iteration process of a snake contour for multiple objects, (b) the enlarged region of the intersection of segments, (c) the splitting and connecting procedure

3.1 Determination of Intersection Segments

In 2-D space, arbitrary two line segments have the relationship of intersecting or non-intersecting. In this section, we discuss how to estimate if two line segments intersect. In order to determine whether two line segments intersect or not, we use vector $\overline{b_n}$ obtained by cross product (for $n = 1,\ 2\ 3,\ or\ 4$).

$$\begin{cases} \overline{b_1} = \overline{S_i} \times \overrightarrow{v_i v_{k-1}}, & \overline{b_2} = \overline{S_i} \times \overrightarrow{v_i v_k} \\ \overline{b_3} = \overline{S_{k-1}} \times \overrightarrow{v_{k-1} v_i}, & \overline{b_4} = \overline{S_{k-1}} \times \overrightarrow{v_{k-1} v_{i+1}} \end{cases} \tag{10}$$

If and only if the starting and ending points of S_{k-1} lie on the same side of s_i, the sign of the z dimension of $\overline{b_n}$ is the same. Otherwise, the z dimension of $\overline{b_n}$ is different. If one of the endpoint of S_{k-1} lies on S_i, for example v_k, the z dimension of $\overline{b_2}$ is zero. The problem of two intersecting line segments can be expressed by the Boolean expression:

$$value = (b_1 \bullet b_2 \le 0 \,\&\&\, b_3 \bullet b_4 < 0) \,||\, (b_1 \bullet b_2 < 0 \,\&\&\, b_3 \bullet b_4 \le 0) \tag{11}$$

where $\bullet$ is inner product, && is AND operator, and || is OR operator.

If $value = 1$, S_i and S_{k-1} intersect.

Otherwise, S_i and S_{k-1} don't intersect.

Equation (11) explains that two line segments intersect only if each of the two pairs of cross products have different signs or one cross product in the pair is zero.

3.2 Splitting and Connecting Contours

If S_i and S_{k-1} intersect, v_k is removed and one contour should split to form two contours. Denote segment vector $\overrightarrow{S_{k-1}}'$ points from v_k to v_{k-1}, which equal to $(-\overrightarrow{S_{k-1}})$, and denote $\overrightarrow{s_{k-1}}'$ as its unit vector as Figure 2 (c) shows. The connecting procedure is decided by:

$$value = (\overrightarrow{s_i} \bullet \overrightarrow{s_k}) < (\overrightarrow{s_i} \bullet \overrightarrow{s_{k-1}}') \tag{12}$$

If $value = 1$, connect snake point v_i with v_{k+1}, v_{i+1} with v_{k-1}. Otherwise, connect v_i with v_{k-1}, v_{i+1} with v_{k+1}.

The new contours are formed after the splitting and connecting operations are performed. A new sequence of snake points of each contour is reorganized by:

$$C_i = \left\{ v_i^b, v_i^1, v_i^2, ..., v_i^{e-1}, v_i^e \mid i < N \right\} \tag{13}$$

where v_i^b and v_i^e are the starting and ending snake points of each contour respectively. N is the total number of contours in the image.

3.3 Algorithm of the Boundary Detection of Multiple Objects

The algorithm of the boundary detection of multiple objects is shown as follows:

Step 1: Convergence process: calculate the energy functions and minimize the energy terms of snake points. If the iteration reaches the final step, stop. Otherwise, go to step 2.

Step 2: The process of determining intersection: if the snake point intersects segment S_i estimated by the equation (11), then go to step 3. Otherwise, go to step 1.

Step 3: The splitting and connecting process: split the contour by removing the unnecessary point v_k. Snake points, which belong to the same side, are connected by equation (12) and then, go to step 4.

Step 4: Reorganizing the sequence of the snake point process: a new sequence is formed for each contour. Go to step 1.

Figure 3 shows the procedure of the proposed method for the detection of the boundaries of multiple objects.

The out-door figures describe how the snake contour handles the topology change when tracking multiple objects. Figure 4 shows that one contour splits into two and Figure 5 presents that two snake contours merge together by using the proposed method.

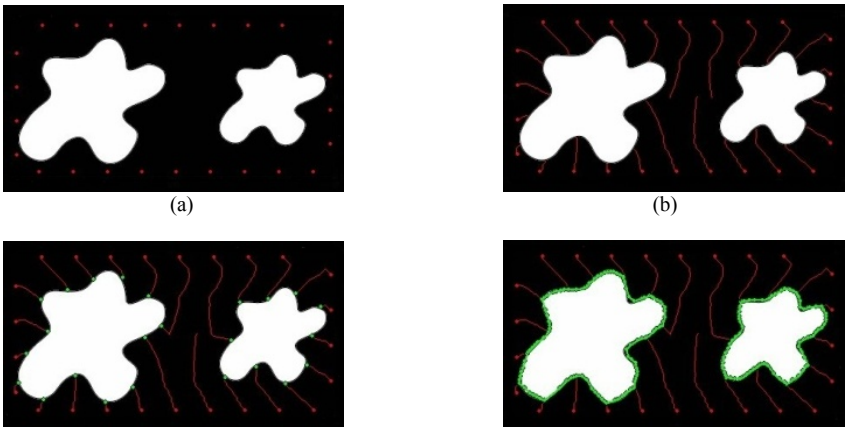


Fig. 3. The procedure of boundary detection of two objects: (a) initial snake contour; (b) the intersection of segments at the middle region; (c) the splitting and connecting process; (d) the boundary detection of multiple objects

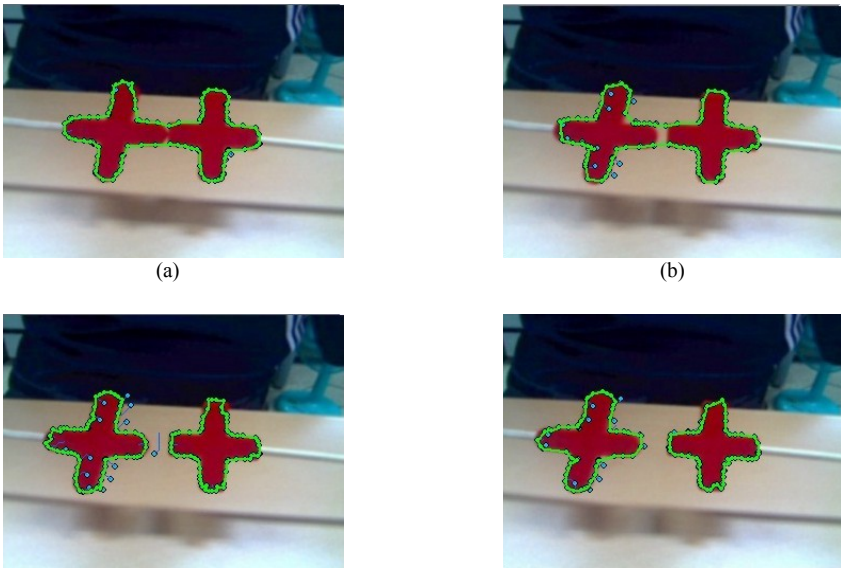
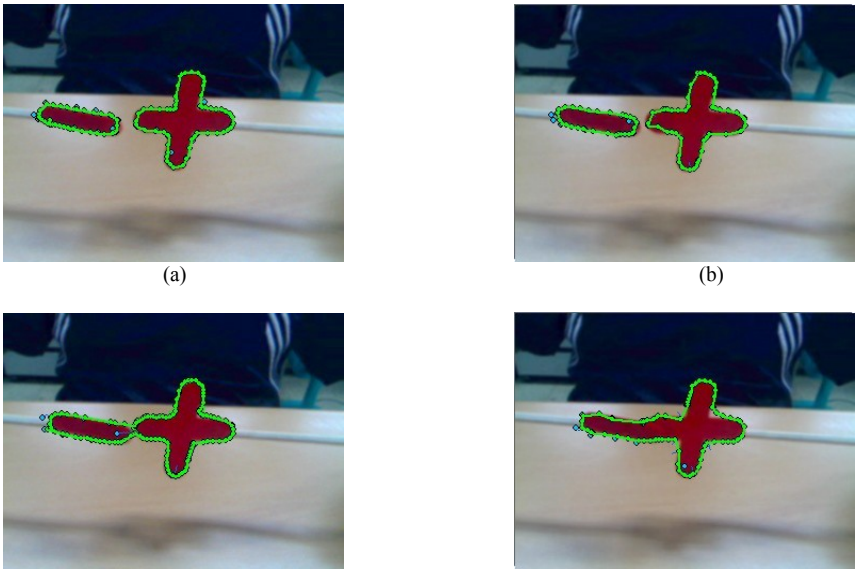


Fig. 4. Two objects are separated in successive 4 frames. $t-1$ frame (marked green) is used as the initialization in the t frame (marked blue)



(a) (b)

(c) (d)

Fig. 5. Two snake contours merge together in successive frames

4. EXPERIMENTAL RESULTS

To verify the performance of our algorithm, a set of experiments was performed. The algorithm was coded in Visual Studio 2005 and the experiments were executed on an Intel Core2 machine running at 1.86GHz with a DDR2 2GB RAM. 320×240 sequence, and images were captured by a Logitech web camera. The size of the synthetic image is 240×240 .

4.1 Performance Comparison of Boundary Detection

The criterion for measuring the accuracy of the performance of the snake in different cases is the degree of similarity between the region of the original object ($R_{original}$) and that of the final convergence of the snake ($R_{estimation}$). To measure this similarity, a *Relative Shape Distortion*, *RSD*(R) is defined as:

$$RSD(R) = \left(\sum_{(x,y)\in f} R_{original}(x,y) \oplus R_{estimation}(x,y) \right) \Big/ \sum_{(x,y)\in f} R_{original}(x,y) \tag{14}$$

where, the $\oplus$ sign is the binary XOR operator. The simulation results are illustrated in Figure 6, and the performance comparison of *RSD* is listed in Table 1.

Figure 6 shows that GVF reached a good result, but this algorithm doesn't deal with the topology changes. So, snake contours have to be initialized around each object manually and it spends a lot of time on convergent iteration. The GAC method is good at implementing topology changes. But with each evolution, it is required to update the values of the level set function for

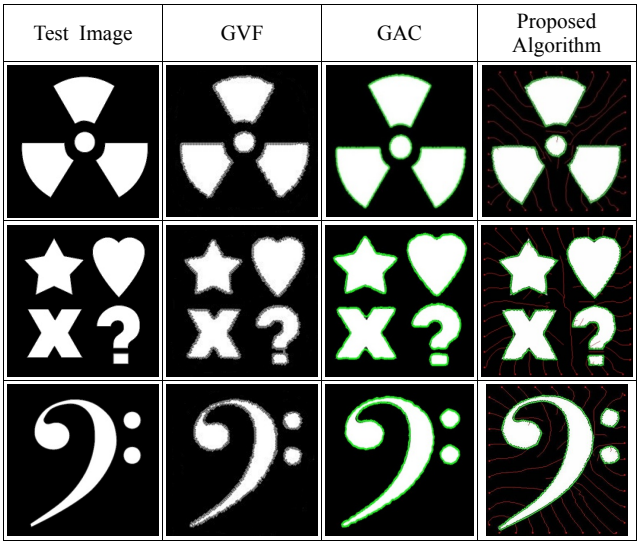


Fig. 6. Performance comparison with the well-known algorithms

Table 1. Performance Comparison

	GVF		GAC		Proposed Algorithm	
	Time	*RSD*	*Time*	*RSD*	*Time*	*RSD*
(1)	15.620s	0.0010	6.700s	0.0007	1.026s	0.0018
(2)	22.553s	0.0018	8.652s	0.0026	1.436s	0.0020
(3)	12.265s	0.0015	5.670s	0.0019	0.735s	0.0020

all the pixels of the whole image. It costs too much computation. Comparing with these two algorithms, our proposed method can get a similar result but can be executed in less time. Performance comparison is shown in Table 1.

4.2 Performance Comparison of Tracking Multiple Objects

Figure 7 illustrates the moment that the white ball strikes the red balls in a snooker game. Different methods are used to track the motion of the red balls. Comparing with the results, the limitations of the GVF and GAC methods are distinctly depicted. Since the GVF method does not provide the implementation of topology changes, it is incapable of implementing it to this kind of problem, even though it has the ability to deal with the single image by manual initialization. In Figure 7 row 2, even though the GAC method handled the topology changes of the snake contour, but part of the contour still passes through the departing balls. This unexpected result is due to the contour that lacks the force to push this part outward along the direction of the motion, the contour can only minimize the energy function by minimizing the length of the contour and (or) the area of the region inside. Obviously, our proposed method not only has the ability of tracking fast moving objects with direction, but its also deals with flexible topology changes. Figure 8 lists more successive frames of the ball's striking moment. The proposed snake contours could track each object separating from the entirety.

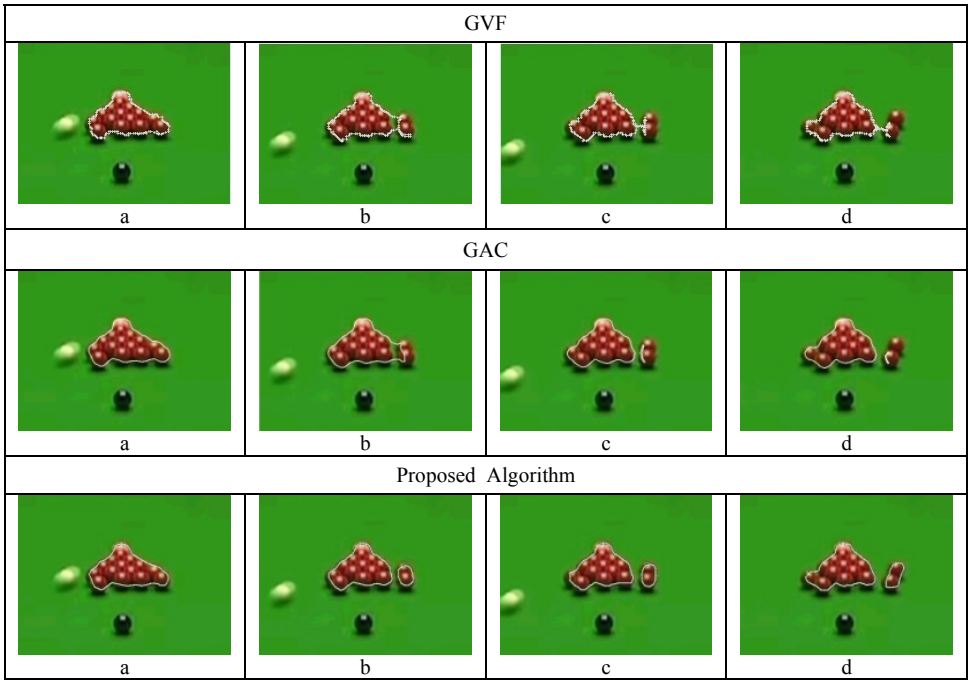


Fig. 7. Performance comparison of tracking snooker balls with the well known algorithms. The successive 4 frames describe the moment when the white ball strokes the red balls. The different snake contours are decided for tracking the red balls. After the right hand two balls are separated from the group, only the proposed algorithm can provide the correct result

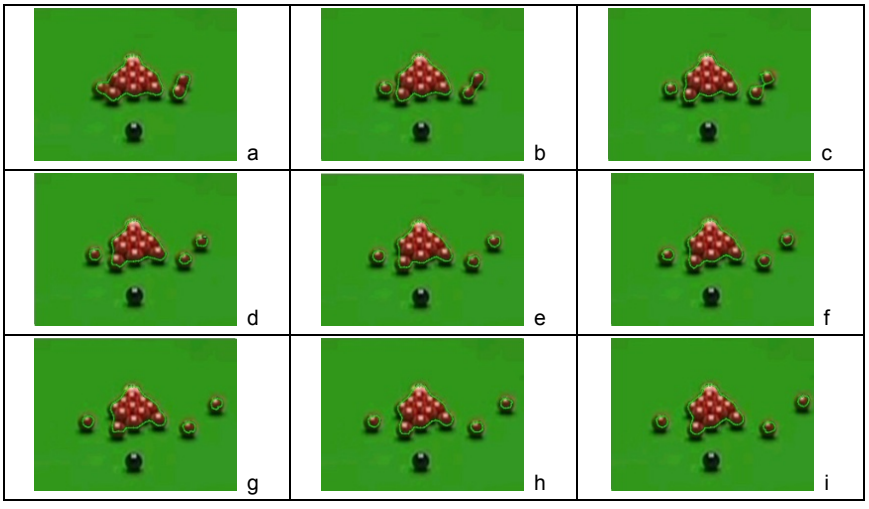


Fig. 8. Tracking snooker balls. After the striking by the white ball, the red balls are scattered to multiple directions. The snake contour deals the topology changes gracefully as it tracks each object

5. CONCLUSIONS

In this paper, we present a new snake algorithm for tracking multiple objects. The proposed snake can be easily split if it encloses more than one object. In order to adapt the tracking of fast moving objects, the new directional energy term improves the ability of snake motion. The results of experiments prove a robust, effective, and accurate performance of the proposed method. In the development of the boundary detection of multiple objects, this proposed algorithm can work in a wide range of applications where the classical active contours have failed.

REFERENCES

[1] M. Kass, A. Witkin, and D. Terzopoulos, "Snake: Active Contour Models," *International Journal of Computer Vision*, Vol.1, No.4, 1987, pp.321-331.

[2] Xu, Chenyang, and J. L. Prince. "*Snakes, Shapes, and Gradient Vector Flow*," IEEE Transaction on Image Processing, Vol.7, No.3, 1998, pp.359-369.

[3] S. H. Kim, A. Alatter, and J. W. Jang. "Snake-Based Contour Detection for Objects with Boundary Concavities," *Optical Engineering*, SPIE, Vol.47, No.3, pp.037002-1 - 037002-7, 2008.

[4] S. H. Kim and J. W. Jang, "Object Contour Tracking Using Snakes in Stereo Image Sequences," *KIPS*, Vol.12-B, No.7, 2005, pp.767-774.

[5] S. S. Yang and H. B. Yoon, "Experimentation and Evaluation of Energy Corrected Snake (ECS) Algorithm for Detection and Tracking the Moving Object," *KIPS*, Vol.16-B, No.4, pp.289-298, 2009.

[6] J. De Vylder, D. Ochoa, W. Philips, L. Chaerle, and D. Van Der Straeten, "Tracking Multiple Objects Using Moving Snakes", 16th IEEE ICDSP 2009, 2009, pp.1- 6.

[7] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic Active Contours". *International Journal of Computer Vision*, Vol.22, 1997, pp.61-79.

[8] T. Chan and L. Vese, "Active Contours Without Edges," *IEEE Transaction on Image Processing*, Vol.10, 2001, pp.266-277.

[9] A. C. Li, C. Xu, C. Gui, and M. D. Fox, "Level Set Evolution Without Re-initialization: A New Variational Formulation," *CVPR 2005*, Vol.1, 2005, pp.430-436.

[10] H. Shan and J. Ma, "Curvelet-Based Geodesic Snake for Image Segmentation with Multiple Objects," *Pattern Recognition Letters*, Vol.31, 2010, pp.355-360.

[11] T. Srinark and C. Kambhamettu, "A Framework for Multiple Snakes and Its Applications", *Pattern Recognition Society*, Vol.39, 2006, pp.1555-1565.

[12] David J. Fleet and Yair Weiss, "*Optical Flow Estimation. In: Mathematical Models in Computer Vision,*" The Handbook, Chater15, Springer, 2005, pp.239-258.

Hua Fang

He received his BS in Computer Engineering from Shandong University, China, in 2003. He received his MS from the Department of Information and Communications Engineering at PaiChai University, Daejeon, Korea, in 2007. He is currently working towards his PhD degree in the same department. His research interests include image processing and computer vision.

JeongWoo Kim

He received his BS in Electronic Physics from the HanKuk University of Foreign Studies in 1997. He received MS degree from the Department of Computer Engineering at PuKyong National University, Busan, Korea, in 2005. He is currently working toward his PhD degree in the Department of Information and Communications Engineering from PaiChai University, Daejeon, Korea. His research interests include image processing, multimedia communications, and computer vision.

JongWhan Jang

He received his BS degree from the Department of Electronic and Communications Engineering from Han Yang University, Seoul, Korea, in 1979. He received his MS and PhD from the Department of Electrical and Computer Engineering from North Carolina State University, Raleigh, NC, USA, in 1986 and 1990, respectively. Since 1990, he has been at PaiChai University, Daejeon, Korea, where he is currently a professor in the Department of Information and Communications Engineering. His research interests include image processing, multimedia communications, and computer vision.