

A NOVEL ACCELERATED GREEDY SNAKE ALGORITHM FOR ACTIVE CONTOURS

Naimul Mefraz Khan and Kaamran Raahemifar

Department of Electrical and Computer Engineering, Ryerson University, Toronto, ON.
e-mail: {n77khan,kraahemi}@ee.ryerson.ca

ABSTRACT

In this paper, we propose a new Accelerated Greedy Snake Algorithm (AGSA) for faster convergence of the active contour optimization problem. The new algorithm takes advantage of the similarity in image pixel gradients to take larger steps in the initial stages of the snake. Due to its fast convergence, the snake can be initialized far away from the object without any issues. This algorithm also uses some intelligent techniques (e.g. re-sampling, relaxation) to maintain a regular shape of the snake at all times while approaching the final contour. Experimental results on three test cases are presented, where the convergence efficiency of our method has been compared with three contemporary algorithms in terms of number of iterations and computational time.

Index Terms— Snake, Greedy, Contour, Optimization

1. INTRODUCTION

Active contours are used in image processing to detect the boundary of an object. They have been widely used in applications like medical imaging, object tracking, video surveillance etc [1]. Before the use of active contours, low level gradient-based techniques like Canny Edge Detector [2] were used for detection of object edge boundaries. But these edge detectors suffer from the problem of discontinuity and are sensitive to noise [3]. On the other hand, active contours provide a smooth and continuous (albeit approximate) boundary. Active contours were first introduced by Kass *et al.* [3] in 1988. In this technique, the object boundary points (pixels) are represented with the help of a parametric curve. The points move around in the image to minimize the *energy function*. The energy function is defined in such a way that the points will reach their minimum energy on the boundary of the object. Since that seminal paper, there has been a lot of research going on, mainly focusing on two aspects of the framework: 1) Convergence the optimization problem and 2) the energy function. The original paper solved the optimization problem with the help of variational calculus. A faster Greedy algorithm was presented in [4]. Variations of the Greedy algorithm has been presented in [5] and [6], where the step sizes and neighborhood patterns of the Greedy algorithm are varied to achieve better convergence. The energy function has also been changed over the time, mainly to fit specific application needs. The Gradient Vector Flow (GVF) snake [7]

provides a significant improvement, where the gradient vectors are introduced over the image to attract the snake to the object accurately, even if the initial snake is far away. This method is one of the first few methods which could attract the snake towards the concavity of an object boundary. However, this method is sometimes avoided due to its slow speed. A faster variation of a similar energy was presented in [8]. A detailed description of the current snake algorithms and their variations can be found in [9].

The primary focus of our research is the convergence aspect of the problem. Our goal is to provide a simple yet robust algorithm to attract the snake towards the object quickly, even if the initial snake is far away. Traditionally, the snake is initialized as a circle with a radius large enough to cover the majority portion of the image. Although there has been research on initializing the snake efficiently [10], most of the special initializations are application specific. The traditional approach is still a subject of interest for its robustness and simplicity. However, with the traditional approach, snakes might result in really slow convergence, specially if the image is of significant resolution.

In this report, we propose a new Accelerated Greedy Snake Algorithm (AGSA) for faster convergence of the snake. Our approach is based on similarity of image pixel gradients in the initial stages of the snake. If gradients have similar value, larger steps can be taken without even considering the energy function value. Our method is based on the Skippy Greedy Snake Algorithm (SGSA) presented in [6]. We also incorporate some intelligent techniques like re-sampling and relaxation to keep the snake in shape while taking larger steps (“accelerating”). Although the experiments presented here are based on the initialization of the snake as a circle, other types of initializations can be incorporated too.

The rest of the paper is organized as follows: Section 2 gives an overview of the active contour model. The Gradient Snake Algorithm (GSA) [4], Fast Greedy Snake Algorithm (FGSA) [5] and the Skippy Greedy Snake Algorithm (SGSA) [6] are also described in this section. Section 3 describes our proposed algorithm, the Accelerated Greedy Snake Algorithm (AGSA). The intelligent techniques incorporated in the method are also described here. Section 4 presents experimental results for three test images, where we compare our method to the GSA, FGSA and the SGSA in terms of num-

ber of iterations and computational time. As we will see from the results, our method provides significant improvement in convergence speed.

2. OVERVIEW OF ACTIVE CONTOUR MODEL

An active contour is modeled as a parametric curve which aims to minimize its internal energy by moving into a local minimum. The position of the snake is given by the parametric curve $v(s) = (x(s), y(s))$. In practice, the snake is considered as a set of control points $v_i, i = 0, 1, \dots, N-1$ which are positioned at an even distance of d from each other. The snake constantly moves under the cost function (defined later) and tries to position itself on the boundary of the object, where it reaches local minima of the function.

The cost function or *energy function* of the snake can be defined as follows:

$$\xi_{snake}(v) = \sum_{i=0}^{N-1} \left[\alpha(\xi_{elast}(v_i)) + \beta(\xi_{curv}(v_i)) + \gamma(\xi_{img}(v_i)) \right]. \quad (1)$$

The Greedy Snake Algorithm works by examining the neighborhood around each snake point and then moving to the position with the lowest energy. Let, each control point examines m neighbors, which are represented by $v_i(j), j = 1, 2, \dots, m$. The energy function defined in Equation (1) consists of three different factors. The first factor $\xi_{elast}(v_i)$ gives a measure of the elasticity and is defined as follows:

$$\xi_{elast}(v_i) = \frac{|\bar{d} - \|v_i - v_{i-1}\||}{\max_j \{|\bar{d} - \|v_i(j) - v_{i-1}\||\}}, \quad (2)$$

where \bar{d} denotes the average distance between all adjacent points. This definition of first order term forces the control points to be equally spaced.

The second factor $\xi_{curv}(v_i)$ in Equation (1) measures the curvature energy and is defined as follows:

$$v_{ss}(s) = \frac{v_{i-1} - 2v_i + v_{i+1}}{d^2}. \quad (3)$$

The parts of the snake where the curve is stretched, the elasticity term will have a high value, while in parts of the snake where the curve is kinked, the curvature term will have a high value. The influence that these two terms have on the overall snake energy is controlled by the control parameters α and β , respectively. These two terms try to give the curve a regular shape by preventing excessive bending or twisting.

The third factor $\xi_{img}(v_i)$ in Equation 1 is the *image energy force*. This term takes into account the image gradient values $-\|\nabla I(x, y)\|^2$, where $I(x, y)$ represents the image intensity values and ∇ is the gradient operator. To eliminate the effect of noise, a Gaussian smoothing operator is usually used as a pre-processing step. While the internal energy tries to keep a regular shape of the contour, the image energy tries to attract the snake towards the object. Since image gradients

provide higher values in case of edge-crossing, this energy compels the snake to stick to the object boundary as much as possible. This term is defined as follows:

$$\xi_{img}(v_i) = \frac{E_{min} - |E|}{E_{max} - E_{min}}, \quad (4)$$

where E represents gradient values, E_{min} and E_{max} denotes the minimum and maximum gradient values in v_i 's local neighborhood (m), respectively. The influence of this term on the overall energy function (Equation (1)) is controlled by the parameter γ .

Overall, GSA results in competitive running times for the convergence of the optimization problem. It also provides some elegant solutions to some of the underlying problems in the original Snake algorithm like strict control parameters, shrinking of the snake etc. (details can be found in [4]). However, since the underlying optimization process is still the simple Greedy algorithm, there is room for improvements. Two extensions of our interest are discussed in the following Section, which mainly focus on faster convergence for the Greedy optimization process.

2.1. The FGSA and SGSA Algorithms

Both of these algorithms were proposed by Lam *et al.* in 1994 and 2006, respectively [5, 6]. These two algorithms use the same energy functions and definitions as defined in the previous section. The only difference is in the neighborhood of a control point being examined.

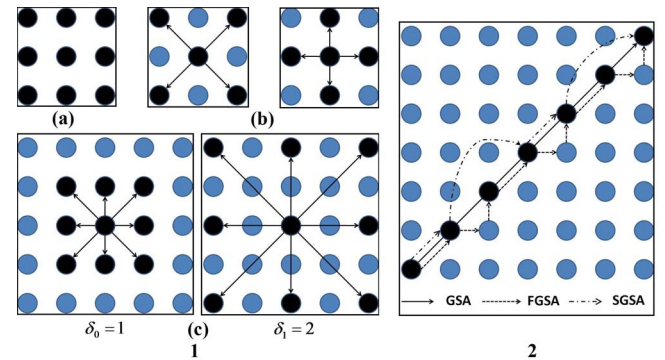


Fig. 1. 1. Different patterns applied by (a) GSA (b) FGSA (c) SGSA. 2. The Optimal Routes for GSA, FGSA and SGSA.

In the Fast Greedy Snake Algorithm (FGSA) [5], instead of applying a fixed pattern like GSA, alternating patterns are used in successive iterations. As can be seen from Figure 1.1(b), there are two patterns, cross-shaped and plus-shaped. Hence, for the GSA, the number of pixels considered for one control point in one iteration is 9, whereas for FGSA it is 5 (assuming a 3X3 neighborhood).

In the Skippy Greedy Snake Algorithm (SGSA) [6], the search is done in two alternating *step sizes*, namely δ_0 and δ_1 . Figure 1.1(c) shows the pixels considered when $\delta_0 = 1$

and $\delta_1 = 2$. When one control point uses step size δ_0 , the neighboring control points use step size δ_1 . These step sizes are switched in every iteration.

Figure 1.2 shows the routes traced by the three algorithms: GSA, FGSA and SGSA. In this figure, it is assumed that the optimal route that a control point can take lies diagonally, from bottom-left to top-right. We can see from Figure 1.2 that GSA takes the straight optimal route, whereas FGSA takes a zigzag route. Hence, it may appear that FGSA is actually slower than GSA. However, since the number of pixels considered is almost half in case of FGSA, the actual running time will be better, as shown in detail in [6]. In case of SGSA, when $\delta_0 = 1$ and $\delta_1 = 2$, no optimal pixels are being skipped. However, SGSA can apply larger δ_1 values in the early stages, when skipping pixels will not overshoot the function. The switch between early stage and “fine-tuning” stage can be made when the number of control points that moved in one iteration is less than a pre-defined threshold.

Although both FGSA and SGSA provide intelligent and efficient alternatives to the Greedy algorithm, and consequently, better running times as we see from the experimental results, there are still some limitations:

1. The effect of taking larger step sizes δ_1 in the earlier stages of convergence as suggested by SGSA is not as useful as it can be, since alternating step sizes are used in neighboring control points to keep the snake in a controlled shape. Even if δ_1 is set to a large value, the neighboring control points still have $\delta_0 = 1$. As a result, the curvature energy term will prevent the control point with large step size to go to the position closest to the object boundary. The alternating control points are assigned large and small step size respectively so that the snake cannot deform beyond repair. This effects the convergence time in turns. Since the experiments provided in [6] uses special initial Snakes which are already close to the object boundary, this effect is negligible in their case. However, in case of a circular initialization, the effect becomes apparent.
2. The switch to the fine-tuning stage ($\delta_0 = 1, \delta_1 = 2$) is made when the number of moving control points are less than a certain threshold. This may result in skipping of important pixels, specially if the shape is not regular and the optimal path is not ideal. One control point may reach near the object boundary quicker than the other control points. And if the switch is not made, this control point may skip over some boundary pixels.
3. As we will see from the experimental results, both in case of SGSA and FGSA, sometimes the control points in the final active contour is not equally spaced. Although the elasticity term defined in Equation (2) tries to force the control points to be equally spaced, the irregular step patterns or sizes overpowers the elasticity

energy. This problem can be solved by paying very close attention to control parameter tuning, which is not always feasible.

Despite these limitations, FGSA and SGSA provide significant improvement over the traditional GSA.

3. THE ACCELERATED GREEDY SNAKE ALGORITHM (AGSA)

In this section, we provide details of our proposed Accelerated Greedy Snake Algorithm (AGSA), which tries to overcome the limitations of the SGSA and FGSA by incorporating some intelligent techniques. The purpose of this algorithm is to provide a robust method, where even with a simple initialization, the snake will converge to the optimal position very quickly. The parameter tuning also does not have to be rigorous.

The principal motivation behind our algorithm was the observation that when a snake is initialized near the image boundary, the underlying pixels in the initial iterations are usually background pixels. In most of the images, background pixels are highly correlated and have similar gradient values. As a result, in the beginning, the optimal position of a snake control point should be the pixel in the control point neighborhood that is closest to the center of the snake, *irrespective of the Energy Function value*. Hence, there is always an attraction towards the center when we are moving through the background pixels. However, since we are ignoring the Energy Function value here, the snake might deform badly. We solve this problem by *re-sampling* the control points before every iteration (explained later). We also incorporate the relaxation technique described in [4] to form sharp corners.

3.1. The Accelerated Step

Let, E_{max} and E_{min} denote the maximum and minimum gradient values in a control point neighborhood. Then:

1. We start by setting δ_1 to a large value.
2. Whenever $E_{max} - E_{min} < C$, we jump to the neighborhood pixel that is closest to the center of the snake, ignoring the energy function value. Here, C denotes the threshold which determines how similar the pixel gradient values should be to decide that they do not represent object boundary.
3. Whenever $E_{max} - E_{min} > C$, we immediately switch back to fine-tuning stage ($\delta_0 = 1, \delta_1 = 2$). The larger difference in gradient values indicates that there might be an underlying area of interest here. In this way, no important pixels will be skipped.

The third step above might be susceptible to noise, since a noisy pixel can generate false gradient value. However, this can easily be overcome by continuously monitoring the value of $E_{max} - E_{min}$ even after switching back to fine-tuning

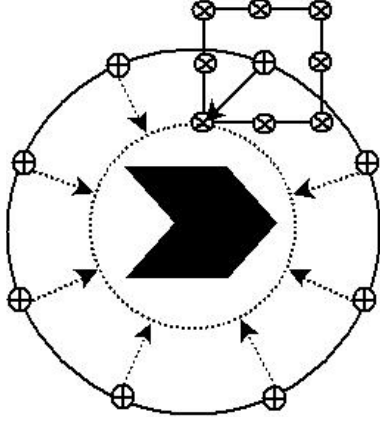


Fig. 2. Effect of the accelerated step of AGSA.

stage. If we find that $E_{max} - E_{min} < C$ for some successive iterations, we can switch back to step 1 again.

Figure 2 depicts the effect of the accelerated step. The neighboring pixels of one control point are shown. As we can see, the control point jumps to the pixel closest to the center of the snake. Similarly, all control points jump to such positions that the snake shrinks towards the center. Eventually, one of the object boundary pixels will come within the range of a control point. Then, we immediately switch back to fine-tuning stage and slowly converge to the final snake. In that way, we will be covering bigger grounds in the initial stages without skipping any important pixels.

3.2. Re-sampling

Since we are ignoring the value of the energy function when accelerating, our control points will move in an irregular way. As a result, they might come too close to each other, or move away too far from each other. To avoid this problem, we use re-sampling before every iteration as follows:

1. Whenever the distance between two successive control points is less than a scalar multiple of the average distance, we remove one of them.
2. Whenever the distance between two successive control points is greater than a scalar multiple of the average distance, we add a new control point in between them.

The scalar multiples above can be varied according to need. In this way, not only does the snake maintain a controlled shape, the final contour consists of evenly spaced control points, as we will see from the experimental results section.

3.3. Relaxation

This idea was first described in [4]. The curvature energy $\xi_{curv}(v_i)$ is re-computed for the control points before every iteration. This re-computation is done to relax the control parameter β to form a sharp corner. Each control point is compared with its two neighbor control points in terms of cur-

vature values. If the curvature value of one control point is higher than its neighbors and higher than a pre-defined threshold, then the magnitude of image gradient at that point is evaluated. If the magnitude is higher than another pre-defined threshold, then for that particular control point β is set to 0. In that way, a sharp corner can be formed at that point [11].

4. EXPERIMENTAL RESULTS

In this section, we provide the results of running the AGSA algorithm on 3 test images. The images were of resolution 200X200, 200X200 and 300X300, respectively. The images with the initial snake (50 control points) are shown in Figure 3(a). The results were compared with that of the GSA, FGSA and SGSA in terms of number of iterations and computational times.

To fairly compare the results, the control parameter values were set to $\alpha = 1.2$, $\beta = 1$, $\gamma = 1.2$. The threshold C for our proposed method was set to 5. Both for SGSA and AGSA, δ_1 was set to 4 in the initial stage. As in [6], the switch to fine-tuning stage in case of SGSA was made when the number of moving control points were less than 10. For the relaxation of β , the curvature threshold was set to 0.3 while the image gradient threshold was set to 120 (please refer to Section 3.3 for details).

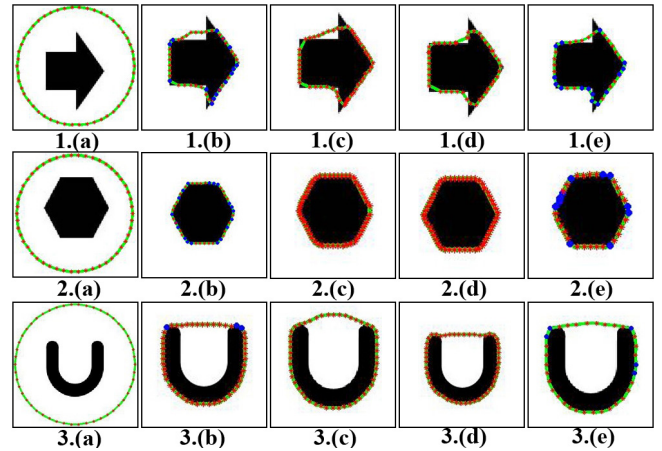


Fig. 3. The original images and resultant images after applying the algorithms: (a) Original image (b) GSA (c) FGSA (d) SGSA (e) AGSA.

Figure 3(b)-(e) shows the final contours achieved for the 3 images by running each algorithm. Table 1 shows the number of iterations needed to converge and the corresponding computational time for each algorithm on each image. As we can see from the results, AGSA reduces the number of iterations to less than half of that of SGSA. Although AGSA incorporates re-sampling and relaxation before each iteration, due to the reduction in number of iterations, the computational time is significantly improved. If we compare the results of GSA and FGSA, we see that although FGSA requires more iter-

Table 1. Comparison of number of iterations and computational time (in seconds) for the four algorithms on the three images (best one bold-faced, second best emphasized).¹

Image:→	Image 1		Image 2		Image 3	
Algorithm↓	I	T	I	T	I	T
GSA	101	3.7	125	4.01	104	3.92
FGSA	134	3.3	148	3.72	136	3.32
SGSA	95	2.8	110	3.21	97	2.99
AGSA	34	1.51	40	1.61	38	1.55

ations, the computational time is reduced. This is because FGSA considers only 5 pixels in the 3×3 neighborhood (Figure 1.1(b)), where GSA considers all the 9 pixels. SGSA and AGSA also considers all the 9 pixels, but due to their skippy nature, the number of iterations and consequently, the computational time is reduced.

We can immediately see the effect of re-sampling and relaxation if we compare the resultant images from Figure 3. The blue dots denote the control points for which β has been relaxed in case of GSA and AGSA. If we compare the result of image 1 for AGSA and SGSA, we see that SGSA indeed has some control points which are too far from each other than the average distance. As a result, the bottom left corner of the final contour is not well defined (Figure 3.1(d)). But in case of AGSA (Figure 3.1(e)), because of re-sampling, the final control points are evenly spaced and the contour is better-defined. In case of AGSA, the final snake for the three images consisted of 36, 35 and 34 control points respectively, reduced from the 50 points because of re-sampling. The corners are also sharper in case of AGSA due to relaxation.

For image 3 (and parts of image 1), we see that none of the algorithms could go into the concave regions of the object boundary. This is because we have not used any gradient vector force in any of these algorithms. The principal purpose of our method was to demonstrate the power of its convergence. Because of the flexibility and simplicity of the method, any gradient vector force [7, 8] can be easily incorporated as a valuable extension to the current method.

5. CONCLUSION AND FUTURE WORKS

A new Accelerated Greedy Snake Algorithm (AGSA) has been presented in this paper. AGSA utilizes the fact that image pixel gradients have similar values in case of background. Hence, larger steps can be taken in the initial stages without even considering the energy function value. To keep the snake in a controlled shape and to achieve a smooth final result, re-sampling and relaxation techniques were incorporated into the algorithm. The proposed method provides better convergence and consequently, reduced computational time even if the initial snake is far away from the object. Our claim is ver-

ified by the provided experimental results, where we compare our method with the GSA, FGSA and SGSA. We see that not only our method provides reduced number of iterations and better computational time, the final snake is smoother with evenly spaced control points because of re-sampling.

In future, our objective is to incorporate some gradient vector energy into the algorithm, so that the final snake can reach the concavity of the objects. We will also focus on coming up with a way to determine the optimal step size and optimal control parameter values efficiently.

6. REFERENCES

- [1] G. Sundaramoorthi, A. Yezzi, A. Mennucci, and G. Sapiro, "New possibilities with sobolev active contours," *International Journal of Computer Vision*, vol. 84, no. 2, pp. 113–129, 2009.
- [2] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.
- [3] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.
- [4] D.J. William and M. Shah, "A fast algorithm for active contours and curvature estimation," *CVGIP: Image Understanding*, vol. 55, no. 1, pp. 14–26, 1992.
- [5] K-M Lam and H Yan, "Fast algorithm for locating head boundaries," *Journal of Electronic Imaging*, vol. 3, no. 4, pp. 352–359, 1994.
- [6] M. Sakalli and K-M. Lam, "A faster converging snake algorithm to locate object boundaries," *IEEE Transactions on Image Processing*, vol. 15, no. 5, pp. 1182–1191, 2006.
- [7] C. Xu and J.L. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Transactions on Image Processing*, vol. 7, no. 2, pp. 359–369, 1988.
- [8] C. Kiser, C. Musial, and P. Sen, "Accelerating active contour algorithms with the gradient diffusion field," *In Proceedings of the International Conference on Pattern Recognition*, 2008.
- [9] R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2010.
- [10] W. Neuenschwander, P. Fua, G. Szekely, and O. Kubler, "Initializing snakes," *In proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 658–663, 1994.
- [11] P. Tiilikainen, *A Comparative Study of Active Contour Snakes*, Copenhagen University, Denmark, 2007.

¹Legend: I → Number of Iterations; T → Computational time (seconds).