Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/081L32

A Parallel Realization of the Active Contour Model on Boundary Extraction

Ping Jiang[1,2,3,*], *Quansheng Dou*[1,3] *and Xiaoying Hu*[4]

[1] School of Computer Science and Technology, Shandong Institute of Business and Technology, Yantai 264005 P. R. China
[2] College of Computer Science and Technology, Jilin University, Changchun 130012 P. R. China
[3] Key Laboratory of Intelligent Information Processing in Universities of Shandong (Shandong Institute of Business and Technology), Yantai 264005 P. R. China
[4] The First Hospital Of Jilin University, Jilin University, Changchun 130012 P. R. China

Received: 8 May. 2013, Revised: 2 Sep. 2013, Accepted: 3 Sep. 2013
Published online: 1 Apr. 2014

Abstract: Active contour is an effective technique for object boundary extraction in biomedical images such as blood vessel tracing or optic disc extraction in retinal images. The traditional serial realization of contour finding needs considerable processing time. This paper presents a parallel realization of the active contour model to speed up the contour convergence process. The initially drawn contour is split into two or more independently controlled sub-contours, and each sub-contour converges independently in parallel. The final true contour is detected by combing results of all converged sub-contours. Compared with the serial realization, the contour convergence in the parallel version is more efficient, and in contrast with other parallel algorithms, the proposed method requires minimal coordination between parallel threads and thus more efficient.

Keywords: Active contour, boundary extraction, converge, sub-contours, parallel threads.

1 Introduction

Active contour [1], also known as snake is an effective technique for object boundary extraction in biomedical images such as blood vessel tracing or optic disc extraction in retinal images. The contours are defined as elastic curves which will converge under the influence of internal and external forces. The internal forces depend on the curve and the external forces are computed from the image. Energy is computed by minimizing a function of the two forces.

Kass, Witkin, and Terzopoulos [1]introduced the concept of an active contour model like energy-minimizing spline guided by external constraint, internal and image forces. Due to the problems of initialization, poor convergence to object boundaries and high time of convergence, a number of methods have been proposed to improve the performance of snakes [2, 3]. Cohen [4,5] introduced a balloon model and enlarged the capture range of snakes. Xu et al. [6] and Xu and Prince [7] proposed a new deformable model called the 'gradient vector flow snake' (GVF). The GVF snake puts emphasis on the problems of short capture range and inability to track at boundary concavity. Improvements have been tried on the original GVF snakes performance. Xu et al. [8] computed the GVF using a polar coordinate representation instead of Cartesian coordinate, and this GVF snake can perform better than the original one in areas of long thin boundary concavities and boundary gaps. Jinyong Cheng et al. [9] improves GVF snake based on wavelet analysis, and particular advantages of the WVF snake over the GVF snake are robust against image noise and the ability to segment the complicated structure of medical images. All the improvements really help the snakes to better capture the object boundaries, which also require a considerable time for processing.

Parallelism is an effective way to speed up the convergence of the contour. Florence Kussener [10] proposed a genetic algorithm to optimize the energy minimization of the snake, and used parallel computing to optimize time of computation. R.M.Curwen, A.Blake and R.Cipolla [11] used B-Spline with L spans to describe the snake, and processed L spans in parallel on a network of transputers, yet the spans need to communicate the effects

* Corresponding author e-mail: ccecping@163.com

of the feature to the rest of the snake. Akiy oshi W akatani [12] divided the contour into parts that contain I control points each and allocated them to different processors which can converge in parallel. It skewed the generation of the DP table and localized the backtrace of the contour, and achieved obvious speed up, yet the parts on different processors are not independent from each other, they need communications between the processors which will delay the convergence process. Fekir A and Benamrane N. [13] proposed a multi-agent system based on NetLogo platform, in which mobile agents move over a grid of stationary agents, each agent represents a point of snake and it minimizes in parallel with other mobile agents, the set of these agents needs to be supervised by Observer.

The main objective of this paper is to propose a simple, efficient and yet effective parallel implementation for the active contour models, which is capable of achieving an accuracy (Ac) similar to the serial version, while providing a faster processing. Developing multithreaded algorithms is a complex problem due to the difficulty of managing and balancing the workload among a large number of threads, as well as synchronizing them [14]. The challenge of deploying a parallel implementation of the active contour model is to keep the amount of communication low. As mentioned in [12], in this paper, the initial contour is also subdivided into sub-contours, yet unlike [12], the sub-contours in our paper are independent from each other, so only a minimal coordination is needed at the final stage of convergence to the object boundary. Our approach assumes a similar intensity distribution pattern across the region surrounding a target boundary. Our experimental results showed that the method improves the converging time without compromising the algorithm Ac.

This paper is organized as follows. In Section 2, we briefly describe the active contour model we used in our implementation. Section 3 outlines the parallel implementation process. Section 4 discusses further smoothness of the contour by B-Spline. Section 5 presents an evaluation of the parallel implementation performance compared with the serial version. Finally, Section 6 concludes with discussion and future work.

2 Active Contour Algorithm

The active contour model is an approach to extract the boundary of an object from a 2D image. It tries to minimize the energy of the current contour which is computed as the sum of the internal and external energy. When the contour converges to the object boundary, its internal energy is minimized, and the most usual approach is to assign low values when the gradient reaches its peak value. When the snake has a shape similar to the object, its external energy should be minimal. The simplest approach is to assign high energy to elongated contours and to high curvature contours.

The snake exhibits dynamic behavior as it always minimizes its energy function. A simple snake which is elastic can be defined as

A.A set of n points
B.An internal elastic energy term
C.An external edge based energy term

To find the contour of the object in an image, we need to initialize the contour near the object and it will iteratively converge to the object boundary. Suppose the active contour is defined by a parametric curve $C(u,t) = [x(u,t),y(u,t)]$, $u \in [0,1]$ where, t determines the temporal position of a point in the sequence, moving through the spatial domain of an image. The energy function of the snake to be minimized can be defined by (1).

$$E_{\text{snake}} = \int_0^1 E_{\text{int}}(C(u)) + E_{\text{image}}(C(u)) + E_{\text{ext}}(C(u))\,du \quad (1)$$

Where, E_{int}, E_{image} and E_{ext} are internal, image and external energy, respectively.

The internal energy defined by (2) depends only on the curve itself, which represents the properties of the quality of the contour: stretching and bending.

$$E_{\text{int}}(C(u)) = 1/2(\alpha|C^{'}(u)|^2 + \beta|C^{''}(u)|^2) \quad (2)$$

$C^{'}(u)$ is the first-order term controls the tension along the curve which is weighted by α and $C^{''}(u)$ is the second-order term controls the rigidity of the curve weighted by β. Set β to zero at a point allows the snake to develop a corner. E_{int} intends to pull or push the curve towards the edges.

The external energy consists of potential forces defined in (3), where E_{image} is defined by the negative gradient of a potential function. The energy is generally the image force as defined in (4) where I denotes the image intensity.

$$E_{\text{external}} = \int_u E_{\text{image}}(C(u))\,du \quad (3)$$

$$E_{\text{image}}(x,y) = -|\nabla I(x,y)|^2. \quad (4)$$

By variational calculus and the Euler-Lagrange differential equation, equation (1) can be solved, and the final contour position can be defined by (5).

$$\alpha C_u - \beta C_{uu} - \nabla E_{\text{image}} = 0 \quad (5)$$

3 Parallel Implementation of Active Contour

As mentioned above, a number of improvements have been made on the active contour model, which also needs more computation time. To accelerate the computation

process, in the paper, we proposed a parallel method to converge the active contour. The method can be combined with most of the active contour approaches mentioned in the literature, because it is independent of how the active contour is modeled and how the minimization process is carried out. Our experimental results with real world images demonstrate that our parallel method can speed up the convergence process.

3.1 Initial Contour Splitting

For the active contour algorithm to find the object boundaries, an initial contour needs to be defined first. As the initial contour is drawn, the region where the object lies in is determined, and we call this region our Region of Interest (ROI). Instead of subdividing the image, we split the contour into two or more sub-contours depending on the shape and complexity of the object. Here for simplicity of the discussion, we split the contour into two sub-contours, and then the two sub-contours start to converge in parallel. It is easy to extend the discussion to four or more sub-contours. The ideal situation is that the two sub-contours complete their convergence simultaneously, so they should have similar sizes.

After the initial contour is drawn, its four coordinates is selected, including the most left, top, right and bottom position, where we denote the x coordinate of the most left and right position as $leftX$ and $rightX$, the y coordinate of the most top and bottom position as $topY$ and $bottomY$. With these four parameters, four points P_1, P_2, P_3,P_4 are created, where $P_1 = (leftX, topY)$, $P_2 = (rightX, topY)$, $P_3 = (leftX, bottomY)$, $P_4 = (rightX, bottomY)$. P_1, P_2, P_3,P_4 define a rectangle surrounding the initial contour, then we split the rectangle into two sub-rectangles with equal size by creating a black line through the center of the rectangle, as a result, the contour is also split into two sub-contours with similar size. In order to make the sub-contours converge to the object boundary, they must be end to end to form a loop, so we used the black line as part of the two sub-contours to help them to form a loop as shown in Fig. 1.

Now we need to get the points of the two sub-contours so that to compute their energies and to converge. When drawing the initial contour, the points are recorded, yet the points on the black line should be extracted to form a loop. The start and end point of the black line can be computed from P_1, P_2, P_3,P_4. The start point $P_{\text{start}} = (leftX, topY + (bottomY - topY)/2)$, and the end point $P_{\text{end}} = (rightX, topY + (bottomY - topY)/2)$. As shown in Fig. 1, according to the positions of P_{start} and P_{end}, we can find the nearest points above and below the black line as the cutting points, then the points of the contour are divided into two sets with each set belonging to one sub-contour, and by adding the points sampled from the black line with equal distance, we finally get two sub-contours.

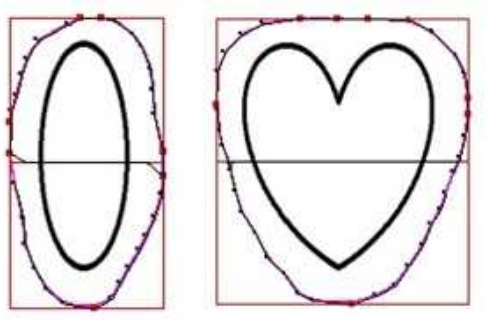

Fig. 1: Splitting the initial contour into two sub-contours by creating a black line through the center of the rectangle

If the object in the image is large or have complex shape, to increase the parallelism, the contour may be split into four parts. Just as described above, the contour is divided into two sub-contours first, then divide the contour vertically by creating another black line from top to bottom, with the start point $P_{\text{start}} = (leftX + (rightX - leftX)/2, topY)$, and the end point $P_{\text{end}} = (leftX + (rightX - leftX)/2, bottomY)$. By the two black lines, the contour is now divided into four sub-contours as shown in Fig. 2, which may be assigned to four different threads to run in parallel.

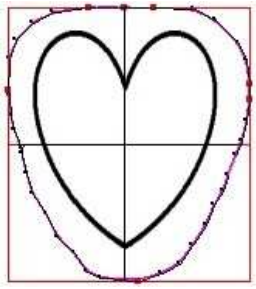

Fig. 2: Splitting the contour into four sub-contours by creating two black lines

3.2 Parallel Implementation

The initial contour is now divided into two sub-contours, which are independent from each other. Then the two sub-contours can converge in parallel. As shown in Fig. 1, the object on the image is also divided into two sub-objects by the black line. On each sub-contour, the points move to the sub-object boundary iteratively, and at each iteration process, all the points are tested one by one. A point will move to a new position if the new position makes the computed energy decrease. So the sub-contour

will converge towards the direction of the energy decreasing. For the points on the black line, because we take the black line as the imaginary boundary of the sub-object, which means that they are already on the boundary, so in our method, we don't move these points at all to save the convergence time. The iteration will end when the energy is minimized, i.e. the sub-contours converge at the sub-object boundaries. Multi-core processors are becoming main stream in computer market due to their high performance, low cost and less power consumption characteristics [15]. We implement the parallelism by Java multithread on a multi-core platform. Each sub-contour is assigned to one thread which then runs in parallel to converge towards the object boundary. Fig. 3 shows the sub-contours during the converging process. Fig. 4 shows the completely converged sub-contours.

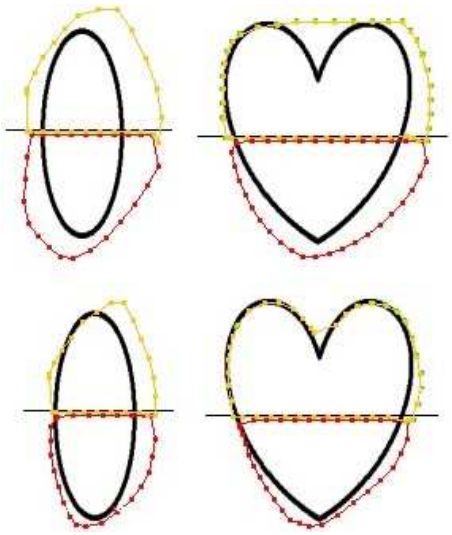

Fig. 3: The Sub-contours during the converging process. The top 2 images show the contours after 10 iterations, the bottom 2 images show the contours after 20 iterations.

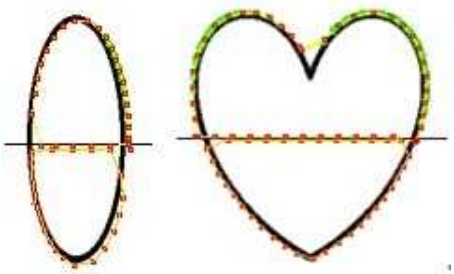

Fig. 4: The two sub-contours after convergence

As mentioned before, the contour is divided into two sub-contours by creating a black line. As a result, more points are added which seems that it will consequently cost more convergence time. Yet, since the points selected from the line don't move at all during the convergence process, therefore the added points have no effect on the convergence time.

3.3 Combine Sub-Contours into the Whole Contour

With the two converged sub-contours, we can get the whole contour of the object by combining them together. First we need to delete the points of the two sub-contours from the black line and combine them, then remove the black line from the image.

A.The pseudo-code to delete the points on the black line and combine the sub-contours is shown in Fig. 5.

```
Ensure: delete the points on the black line, combine the
sub-contours into the whole contour.
for i = 0 to P1.size() − 1
        p = P1.get(i);
        if p is on the black line,
            put p into Pt1;
            delete p from P1;
        end if;
end for;
for i = 0 to P2.size() − 1
        p = P2.get(i);
        if p is on the black line,
            put p into Pt2;
            delete p from P2;
        end if;
end for;
for i = 0 to Pt1.size() − 1
      select the most left point p1left;
      select the most right point p1right;
end for;
for i = 0 to Pt2.size() − 1
      select the most left point p2left;
      select the most right point p2right;
end for;
for i = 0 to P1.size() − 1
  select the closest neighbour points of p1left and
p1right in P1 written as pn1left, pn1right;
end for;
for i = 0 to P2.size() − 1
  select the closest neighbour points of p2left and
p2right in P2 written as pn2left, pn2right;
```

Fig. 5: the pseudo-code to delete the points on the black line and combine the sub-contours

B.Remove the black line

As mentioned before, when splitting the initial contour, a black line is created in the way as described

in the following. The pixels of the image containing the object can be extracted and stored in an array *imageArray*, through the start point P_{start} and the end point P_{end}, a line can be defined, and the pixels on the line are stored in an array *pixArray*, then we set the pixels on the line to be black, as a result, a black line is created.

When the object contour is found, the black line should be removed, i.e. the colour of the pixels on the black line should be restored to the original values. So we just copy the colour values from *pixArray* to the corresponding part of the *imageArray* determined by P_{start} and P_{end}. Finally we get the contour surrounding the object. Fig. 6 shows the results after the combination.

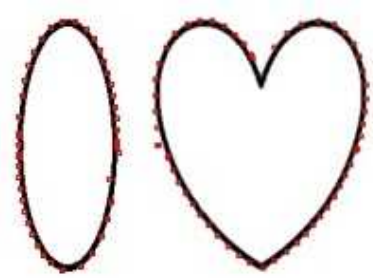

Fig. 6: The Contours after removing the black bine

4 B-spline Adjustment

After combining the sub-contours, now we get the whole contour of the object. The main application of our method is on the processing of the retinal images. The detection of accurate boundary of the optic disc is important for the detection and diagnosis of Glaucoma where the variation in the shape and size of the optic disk is used to detect and measure the severity of disease [16]. As shown in Fig. 7, in retinal images, the optic disc (OD) segmentation is a challenging task mainly due to blood vessel occlusions, illdefined boundaries, image variations near disk boundaries due to pathological changes and variable imaging conditions [17]. The contour is often not as closely to the object boundary as shown in the above figures.

From Fig. 7 we find that only a small segment of the contour is apart from the object boundary. So we only need to adjust this segment while leaving the others unchanged. B-Spline is a perfect choice for local adjustment. The term "B-spline" was coined by Isaac Jacob Schoenberg and is short for basis spline [18]. In this paper, we take advantage of its locality property which indicates that moving a control vertex will change at most K curve segments, where K is the order.

Fig. 7: The contour of the optic disc in a retinal image

Give m real values t_i, called knots, with $t_0 \leq t_1 \leq \ldots \leq t_{m-1}$, A B-Spline of degree n is a parametric curve composed of a linear combination of basis B-Splines $b_{i,n}$ of degree n

$$S(t) = \sum_{i=0}^{m-n-2} P_i B_{i,n}(t), t \in [t_n, t_{m-n-1}].$$

The points $P_i \in \mathbb{R}^d$ are called control points. There are $m-n-1$ control points.

In our case, the contour is transformed into a cubic B-Spline whose control points are the points on the contour. The cubic B-Spline curve segment is determined by (6):

$$P_{0,3}(t) = \frac{1}{6}[1\ t\ t^2\ t^3] \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \quad t \in [0,1] \quad (6)$$

Where, P_0, P_1, P_2, P_4 are control points on the contour.

By transforming the contour shown in Fig. 7 into B-Spline, we get the result as shown in Fig. 8.

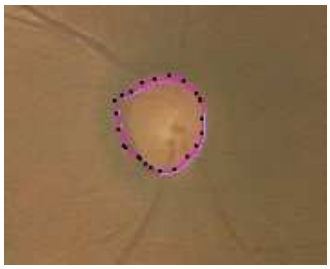

Fig. 8: B-Spline generated from the contour

As shown in Fig. 8, the segments at the left side of the optic disc are apart from its boundary, so we just need to move the control points of these segments to adjust the Spline to the boundary. The final contour after adjustment is shown in Fig. 9:

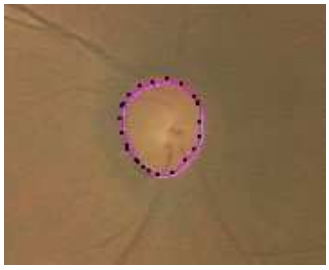

Fig. 9: The final B-Spline contour after local adjustment

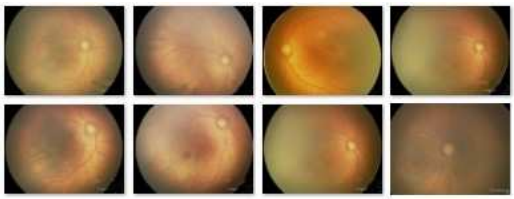

Fig. 10: The left is the 6 images containing typical objects. The right is the retinal image on which the contour of the optic disc is to be located

5 Evaluation

Modern CPUs integrate multiple cores and provide hardware support for parallel processing [19]. Multicore and manycore processors have become the standard building block for desktop and scalable computers [20]. So our parallel implementation was developed and tested on Windows 7 platform, with Intel(R) Core(TM)2 Duo CPU E7600 @3.06GHZ 3.06GHZ and 4GB RAM. The parallelism of active contour model is based on Java multithread. However, developing multithreaded algorithms is a complex problem due to the difficulty of managing and balancing the workload among a large number of threads, as well as synchronizing them [21]. Because the sub-contours are totally independent from each other, and the splitting process ensures the similar size of the sub-contours, so the difficulty of multithreaded algorithms is solved. Depending on the shape and complexity of the object, the initial contour can be divided into two, four or even more sub-contours, by taking advantage of the multithread-based parallel computing in Java, each sub-contour is assigned to a thread which can run in parallel on a different core, and then these sub-contours can converge to the sub-object boundaries simultaneously.

The previous serial implementation was developed by Java language. The initial contour converges iteratively to the object boundary by moving one point each time. While in the parallel version, the sub-contours converge simultaneously (as described in Section 3). The challenge of deploying a parallel implementation is to keep the amount of communication low. In this paper, the sub-contours are completely independent from each other, so they don't need any communication. When the two sub-contours end their convergence, they need to be combined to form the whole contour, which is the only reason for delay.

We selected 8 retinal images as our test images to evaluate the serial and parallel active contour implementation. The 8 images are shown in Fig. 10.We tried to find the contour of the optic disc. Table 1 shows the execution time performance of the serial and parallel version. Fig. 11 shows the contour of the optic disc in Img8. In order to make the result easy to show, we cut Img8 so as to focus on the optic disc.

Table 1: Execution times (in seconds) of serial and parallel implementation

Image	Serial Version	Parallel Version	Speedup
Img1	1.885	1.094	1.723
Img2	2.347	1.418	1.674
Img3	2.239	1.564	1.432
Img4	3.078	1.896	1.623
Img5	2.842	1.787	1.590
Img6	2.478	1.422	1.743
Img7	2.713	1.696	1.600
Img8	1.975	1.113	1.774

From table 1, we find that the convergence has been accelerated by a speedup of 1.645 on average as the contour split into two sub-contours.

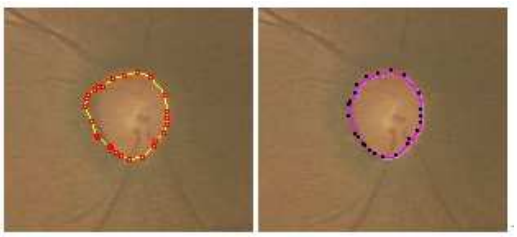

Fig. 11: The final contour of the optic disc in retinal image. The left image shows the contour converged by the software itself. The right image shows the B-Spline contour adjusted by user.

6 Conclusion

This paper introduced a parallel algorithm for the active contour finding algorithm. The implementation is based on the contour splitting, which achieves a faster convergence of the target contour. Depending on the shape and complexity of the object, the contour can be split into two, four or even more sub-contours. The sub-contours are assigned to different threads which can run in parallel, and then these sub-contours can converge

to the sub-object boundaries simultaneously. Finally the converged sub-contours are combined together to get the whole contour of the object. Our further study will generalize the method to the detection of contour with significant variability of image intensity distribution near the target boundary.

Acknowledgement

This work was partially supported by the Young Foundation, under grant No. 2011QN083, National Natural Science Foundation of China, under grant No. 61272244,60970088.

References

[1] Kass,M., Witkin, M. and Terzopoulos, D., Snakes: active contour models. International Journal of Vision, **1**, 321-331 (1987).
[2] McInerney, T. and Terzopoulos, D., Deformable models in medical image analysis: A survey. Medical Image Analysis, **1**, 91-108 (1996).
[3] Bamford, P. and Lovell, B., Unsupervised cell nucleus segmentation with active contours. Signal Processing Special Issue: Deformable Models and Techniques for Image and Signal Processing, **71**, 203 - 213 (1998).
[4] Cohen, L., On active contour models and balloons. Computer Vision, Graphics and Image Processing: Image Understanding, **53**, 211-218 (1989).
[5] Cohen, L. and Cohen, L., Finite-element methods for active contour models and balloons for 2-d and 3-d images. IEEE Transactions on Pattern Analysis and Machine Intelligence, **15**, 1146- 1131 (1993).
[6] Xu, C., Pham, D., Rettmann, M., Yu, D. and Prince, J., Reconstruction of the human cerebral cortex from magnetic resonance images. IEEE Transactions on Medical Imaging, **18**, 467- 479 (1999).
[7] Xu, C. and Prince, J., Generalized gradient vector flow external forces for active contours. Signal Processing, **71**, 131-139 (1998).
[8] Yu, Z. and Bajaj, C., Image segmentation using gradient vector diffusion and region merging. In ICPR'02, Quebec City, 828-831 (2001).
[9] Jinyong Cheng, Yihui Liu, Ruixiang Jia and Weiyu Guo, A new active contour model for medical image analysis-wavelet vector flow. In IAENG International Journal of Applied Mathematics, Hong Kong, March 21-23 (2007).
[10] Florence Kussener, Active contour: a parallel genetic algorithm approach. In ICSI 2011, Chongqing, China. June 12-15, (2011).
[11] R. M.Curwen, A.Blake and R.Cipolla., Parallel implementation of Lagrangian dynamics for real-time snakes. In Proc. British Machine Vision Conference, 29-35 (1991).
[12] Akiy oshi W akatani., A scalable parallel algorithm for the extraction of active contour. In Proc. PARELEC 2000. Trois-Rivieres, Que., Canada, 94-98. August 27-30, (2000).
[13] Fekir A, Benamrane N. Segmentation of medical image sequence by parallel active contour. In Adv Exp Med Biol, **696**, 515-22 (2011).
[14] Carlos Ordonez, Mario Navas and Carlos Garcia-Alvarado. Parallel multithreaded processing for data set summarization on multicore CPUs. In JCSE, **5**, 111-120 (2011).
[15] Benhai zhou, Jianzhong Qiao, Shukuan Lin, Research on parallel real-time scheduling algorithm of hybrid parameter tasks on multi-core platform, Applied Mathematics & Information Sciences, **5-2S**, 211S-217S (2011).
[16] Siddalingaswamy P. C. and Gopalakrishna Prabhu .K. Automatic localization and boundary detection of optic disc using implicit active contours. International Journal of Computer Applications (0975-8887), **1**, (2010).
[17] Gopal Datt Joshi, Rohit Gautam and Jayanthi Sivaswamy. Robust optic disk segmentation from colour retinal images. In ICVGIP'10, December 12-15, (2010).
[18] Carl de Boor. A Practical Guide to Splines. Springer-Verlag, 113-114 (1978).
[19] Konstantinos Krikellas, Marcelo Cintra and Stratis D. Viglas. Multithreaded query execution on multicore processors. In VLDB, (2009).
[20] Christopher G. Baker, Michael A. Heroux and H. Carter Edwards, Alan B. Williams. A Light-weight API for Portable Multicore Programming. In 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, 601-606 (2010).
[21] Carlos Ordonez, Mario Navas, and Carlos Garcia-Alvarado. Parallel Multithreaded Processing for Data Set Summarization on Multicore CPUs. Journal of Computing Science and Engineering, **5**, 111-120 (2011).

Ping Jiang works as an instructor in School of Computer Science and Technology, Shandong Institute of Business and Technology. He is a Ph.D. student at Jilin University. His research interest covers biomedical image processing, artificial intelligence and biological information calculation etc. He is the Corresponding author of this paper.

Quansheng Dou Ph.D. works at School of Computer Science and Technology, Shandong Institute of Business and Technology as an associate professor. His research interest covers Intelligent scientific theory and method ,data mining, multi agent technology, bionic calculation and swarm intelligence etc.

Xiaoying Hu Ph.D, works at the first hospital of Jilin University as a professor. She is good at diagnosis and threatment of glaucoma,and radiography and laser treatment.