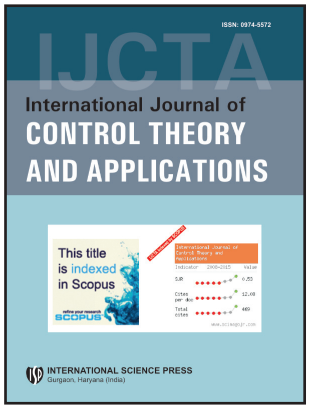

International Journal of Control Theory and Applications

ISSN : 0974-5572

© International Science Press

Volume 10 • Number 12 • 2017

Parallelization of Deformable Models in Multicore Architecture

Priya P. Sajan[a] **and S.S. Kumar**[b]

[a]*Research Scholar, Noorul Islam University Noorul Islam Centre for Higher Education Kumaracoil, Thuckalay*
[b]*Research Guide, Associate Professor Noorul Islam University, Noorul Islam Centre for Higher Education Kumaracoil, Thuckalay*

Abstract: OpenMP is a shared memory API whose features are based on prior efforts to facilitate shared memory parallel programming. It is highly cost effective, easy to implement with lesser modifications in existing code and could run effectively in many different platforms. Since OpenMP emphasis on structured parallel programming, it could be effectively adapted in digital image segmentation. GVFActive Contour Snakes is one such segmentation method that exhibits structured parallelism. This method can specifically locate convergent concavities of object boundary. This is obtained by diffusing the gradient vectors of an image and by the influence of external forces computed within the region of interest. But this process involves complex mathematical computations which consume huge execution time. This huge processing time can be immensely reduced with the parallel programming constructs of OpenMP.

Keywords: GVF, Active Contour Snake, OpenMP, API.

1. INTRODUCTION

Todays multicore processing architecture demands simple parallel application development, which OpenMP provides with language extensions and tools. The appropriate insertion of OpenMP compiler directives features sequential program to its parallel version so that applications can benefit from shared memory parallel multicore architecture, often with minimal modification of existing algorithm. Deformable Active Contour Snake is one such method which possess parallelism on instruction level, data level and loop level. GVF requires prior object information such as shape, approximate co-ordinate location and size of the object for improving the accuracy of segmentation in accordance with specific region of interest.

The success of OpenMP can be attributed to a number of factors. One among is its strong emphasis on structured parallel programming. Another is that OpenMP is comparatively easy to use and be implemented to the existing method, since the burden of working out the details of parallel program is up to the compiler. It has the major advantage of being widely adopted, so that an OpenMP application can run on multiple platforms.

The first step in creating an OpenMP program from a sequential one is to identify the parallelism it contains, which means finding instructions, sequences of instructions or even large regions of code that may be executed concurrently by different multiple processing cores sharing a common memory. To attain this, portions of code has to be re-organised to obtain independent instruction sequences. It may even be necessary to replace an algorithm with an alternative parallel version that accomplish same task, but offers more exploitable parallelism which results processing within less time. The second step is to express the specification type of the parallelism that has been acquired through dividing and splitting the iteration set. The huge practical benefit of OpenMP is that it can be applied to incrementally create a parallel program from an existing sequential code. It is possible to insert directives into a portion of the program which exhibits instruction, data or loop level parallelism. After attaining the expected speed and accuracy, the parallel processing could be terminated by applying suitable directives in the program.

Gradient Vector Flow is one such method which exhibits structured parallelism in various aspects of instruction level, data level and in loop level. GVF method which is in sequential manner can be converted to exhibit parallelism by applying appropriate OpenMP directives so as to exploit the processing power of multicore architecture. Implementation of GVF with OpenMP represents a coarse grain parallel approach in which multiple processing threads are created simultaneously for computing the region of interest which is supposed to be segmented.

2. RELATED WORKS

There had been a lot of research made on the parallelization of GVF Snake Active Contour method on GPU cards. In [1], Rigo, Juan and Julio made parallel segmentation of medical images with deformable models on GPU with CUDA and OpenMP. Hybrid image classification and parameter selection using a shared memory parallel algorithm was made by Philips, Watson and Wynein [2]. Mahmoud and Jumaily in [3], conducted the segmentation of skin cancer images based on GVF snake. A comparative study on GPU programming models is done in [5] by Pallippuram and Bhuiyan. Application of OpenMP and its various strategies is well described by Chapman, Jost and Van Der Pas in [8].

In [9], He Z and Kuester made an attempt to parallelize GVF using OpenGL in GPU environment. In [10], using GPU, Zheng and Zhang used texture arrays for the calculation of GVF field and subsequent iterative computations. Perrot implemented a segmentation algorithm on GPU which is based on statistical deformable model. Also in the GPU environment, researches were done for implementing GVF based on Markov Random Fields (MRF) [16], tile approaches [14] etc. The resultant output with these implementations is promising, but their architectural cost and programming complexity both in terms of time and space are very high. There had been only fewer studies conducted regarding implementation of image segmentation methods in shared memory architecture using OpenMP. This method is highly cost effective and easy to implement in existing platform with less modification.

3. DEFORMABLE MODELS

Deformable models also known as Active Contours is the widely accepted and commonly used region based model technique which takes advantage of prior knowledge concerning the position and form of anatomical structures. The mostly used representation of deformable models is the so called snake, which uses a-priori knowledge by definition and parameterisation of an energy term as well as by the placement of the initial model in the images. Deformable models are differential equations that give direction to fix conclusively the shape, directions and movement of curves reinforced of an abstract elastic material. The physical interpolation of a deformable model is that of an elastic body that responds to the forces applied on it.

There are basically two types of deformable models: parametric deformable models and geometric deformable models. Parametric deformable models represent curves and surfaces explicitly in their parametric forms during deformation. This representation allows direct interaction with the model and can lead to a compact representation for fast real-time implementation. GVF Snake is a parametric deformable model that adapts model topology for splitting or merging parts during deformation.

4. GRADIENT VECTOR FLOW

Gradient Vector Flow provides an advancement to the traditional parameterized snake $X(s) = (X(s), Y(s))$, $s \in [0, 1]$, that minimizes the energy functional. Gradient Vector Flow overcomes the difficulties of less capture range and convergence in boundary concavities by applying distance potential force which is a dense vector field that is derived from an image by minimizing energy functional coefficients.

As compared with the traditional snake models, GVF Active Contour Snakes can be initialized by assigning an initial seed value to start up the computation inside, throughout or across the edge boundary, or outside the area of interest. Reason behind this mode of operation is because Gradient Vector Field is the resultant outcome of an isotropic diffusion process that may not disrupt or blur the edges within the image when compared with Gaussian filters.

(a) Image to be segmented **(b) segmented region with proposed parallel GVF**

Figure 1:

The basic step in computing the Gradient Vector field is to get the contour map $f(x, y)$ which is obtained from the input image $I_M(x, y)$. The contour map is a potential function which is defined over the image plane whose value will be slightly larger nearby the edges of interested area which is proposed to be segmented.

The capture range of the produced field is directly proportional to the number of iterations and hence determine the GVF field. This way of controlling the capture range enables the GVF method to eliminate the location dependancy for the initial active contour. If a certain shape of interest is not included or is partially included in the initial snake, the GVF field will allow deformations of the initial active contour to include the shape. If the region of interst is located completely out of the initial active contour, it is even possible that the complete snake move towards the shape to capture it, provided there are no additional shapes in the neighbourhood.

The main drawback of GVF active contour snake is the involvement of huge and complex mathematical computation which worsely increases the processing time. No matter how faster the computers are, technology have to be developed to utilize this high power processing hardware architecture. The computers with multicore architecture means computers with multiple functional units that may be able to operate simultaneously. To accomplish this compilers are developed to determine the dependencies between operations and their orderliness which may exploit instruction and data level parallelism in the subjected area thereby utilising parallel shared memory architecture.

In the past few decades, technological progress has made it possible that the rate at which instructions are executed are too fast. Since multicore technology is going mainstream, it is vital that application software must be able to make effective use of parallelism that is present in todays hardware architecture.

5. PROPOSED PARALLEL GVF SNAKE ALGORITHM WITH OPENMP

First, gray level scale luminance of the actual image is computed. For this a weighted sum must be calculated in a linear RGB space thereafter by removing this Gama compression function through Gama expansion.

$$\{(\text{———})$$

The Gradient of image is calculated for each pixel position in the image. Then the pixels are partitioned depending on their intensity value. Distance between pixel map is calculated and then converted to binarized gradient to show and display the gradient flow value of the pixels in the region of interest. In the proposed segmented area, the initial seed points are calculated based on the binarized gradient value.

This sequential procedure can be converted to an OpenMP parallel construct by applying and adopting any or all of the following means.

(a) Create teams of threads for parallel execution

(b) Specify how to share work among the thread team members

(c) Declare both shared and private variables

(d) Synchronize threads and enable them to perform certain operations exclusively without interference of other co-processing threads

Proposed parallel algorithm for GVF Active Contour Snake can be summarised with the following flow chart.

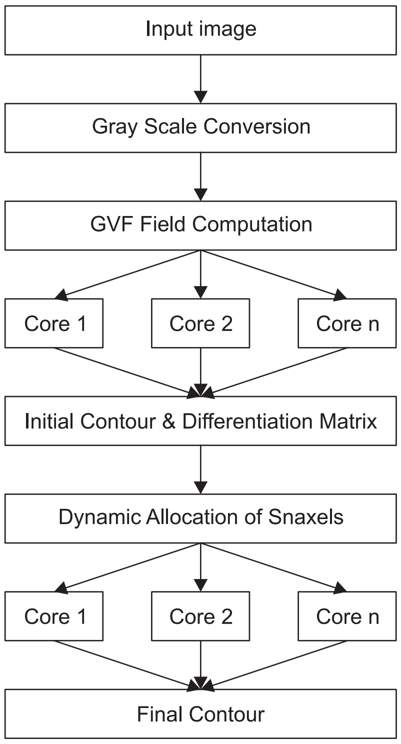

Figure 2: Proposed Parallel GVF Model

Execution platform ie in the multicore system, input image will be evenly dispersed into multiple processing threads using suitable pragma directives. After calculating the gradient pixel values, the *omp parallel* construct is used to perform the computations in parallel manner. Parts of the program that are not enclosed by a parallel construct will execute sequentially. When a thread encounters this construct, a team of threads is created to execute the associated parallel region. At the end of the parallel region, there is an implied *barrier* that forces all the threads to wait until the work inside the region is completed. Only the initial thread continues execution after the end of the parallel region. The thread that encounters the parallel construct will become the *master* of the new team. Each thread in the team is assigned a unique thread number which may range from zero up to one less than the number of threads within the team.

While mapping the distance between the pixels with similar gradient level, data level parallelism can be applied with *omp critical* construct. This provides a means to ensure that multiple threads do not attempt to update the same shared thread simultaneously. Hence there will not be a risk that multiple updation occurs and when a thread encounters a critical region, it waits until no other thread is executing in the same region with same name. The OpenMP implementation adopts instruction, data and loop level parallelism so that hundreds of pixels in the GVF field will be computed simultaneously in a parallel manner so that the processing speed and efficiency will be steeply increased.

For calculating the gradient value of the input image, the RGB value is converted into grayscale which will be stored in an array for further processing. Threshold value for each pixel will be calculated based on the internal energy values. This will be a weighted sum which determines the gradient value for each pixel position. The pixels will then be classified depending on this gradient intensity value. In a normal sequential scenario this a time consuming process because the intensity value of each pixel in the gradient field has to be individually processed and calculated. Using OpenMP, the task of conversion of RGB to grayscale, calculation of threshold value can be divided among the multiple processing cores using work sharing directives *#pragma omp for* and *#pragma omp parallel*. Pseudocode for this processing is summarized below.

```
#pragma omp parallel shared(n)
{ for(i=0;i<=n;i++)
    #pragma omp master
    1.   compute length of contour region
    2.   compute the rgb values to gray scale
    #pragma omp for private
    1.   calculate each point of contour region
    #pragma omp for nowait()
    1.   boundary check with pixel allocation for GVF field computation
    #pragma omp for atomic nowait()
    1.   initial contour identification
    2.   dynamic allocation of snaxels
    #pragma omp for ordered()
    1.   final contour extraction
}
```

A parallel region is active if it is executed by a team of threads consisting of more than one thread. The work sharing constructs specifies the region of code where work is to be distributed among executing threads and the manner in which it to be performed. The outer for loop in the RGB gray scale is parallelized by giving

#pragma omp parallel according to the number of threads set created. The *master* construct defines a block of code that is guaranteed to be executed by the master thrad only and has an implied barrier on entry and exit. This *master* region initializes the contour region and is distributed to compiler generated threads. The advantage of *master* region is that it doesn't implies *barrier* regions such that each threads can process simultaneously without awaiting the result generated by coprocessing threads. Executing the problem using 8 threads in an 8core CPU, the processing time will be reduced by one eighth. The thread number is directly proportional to the number of processing core and hence by increasing the threads numerically in the program will not be having much impact due to the increased time to fork and join.

For calculating the gradient value and obtaining its corresponding intensity, two *for* loops are required. The first *for* loop shares the parameters for contour generation in *private* manner and the second *for* loop as *nowait.* Practically it may requires additional computation after the *master* construct been initialized. Instead of making the inner loop parallel, the outer loop is the better choice. Another *omp* directive *private* is also used here because it directs the compiler to make variables private so that multiple copies of the same variable will not execute again and again. By adopting this mechanism, it is seen that the execution time again reduced to one eighth when compared with the same sequential mode and when executed in an 8 core system.

6. IMPLEMENTATION OF PARALLEL GVF SNAKE WITH OPENMP

The experiments have been conducted in two architectures: one in Intel Core i7-3770 CPU at 3.40GHz and 8GB RAM with 64 bit Operating System and the second one in Intel Core i3-E7500 at 2.93GHz with 32 bit Operating System. For illustrative purpose both medical images as well as miscellaneous images have been tried. Size of those images ranges from 128x128 to 2048x2048. The implementation time for both sequential and parallel version of the algorithm is presented as an inference of adopting parallelism in multicore systems. All the results presented are measured in terms of seconds and represented in single precision floating point numbers. The same initial contour is used for the entire images. All the medical images used in the experiment are 8 bits per pixel and for the miscellaneous images, it is 16 bits per pixel.

The program was able to execute in both Windows and Linux platforms. The sequential program is written in Matlab, Java and C language. Further the parallelization of code was made by implementing OpenMP directives in the C version of the program. Again this parallelized code is made executed in both Windows and Linux platforms to make a comparative study on processing speed and efficiency in the kernel level, if any. This kernel-thread communication governs the behaviour of program at run time. This message passing can be made effective through OpenMP functions and the mode by which environment variables are shared and accessed. Library routines can also be used to give the values to control the variables as they can override the predefined values. The experiment showed almost similar execution time for the C version of the OpenMP when it made executed in Windows and Linux and has an immense processing speed and efficiency when compared with the traditional sequential program.

7. PERFORMANCE EVALUATION

In order to evaluate the performance, time required for the execution of both sequential and parallel version of the algorithm were experimented. Sequential version was executed in Windows 4 core and 8 core systems with Matlab, Java and C; also parallel version using OpenMP with C language both in Wndows and Linux platforms. The segmented result and time comparison obtained by executing the program in quad core and 8 core system is summarised in the following two tables.. The entire processing time for the proposed parallel version of the GVF snake method may vary with the following artifacts:

(i) number of processing cores

(ii) number of compiler threads created

(ii) mode by which the variables are shared and accessed

As the number of processing cores increases, processing efficiency will also increases there by reducing the execution time. Also the number of compiler threads created automatically is directly proportional to the number of cores. Inspite of these threads, programmer can create light weight processes called threads. The *num_threads* class in OpenMP supports this parallel construct and can be used to specify how many threads should be in the team executing the parallel region.

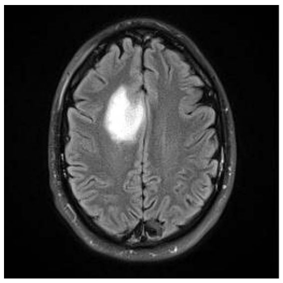
Figure 3: (a) Image 1

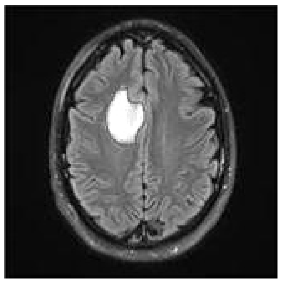
Figure 3: (b) Image 1

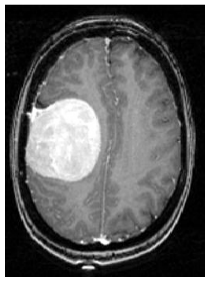
Figure 4: (a) Image 2

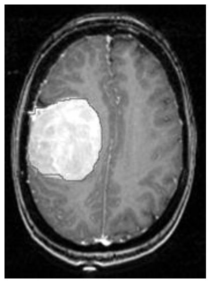
Figure 4: (b) Image 2

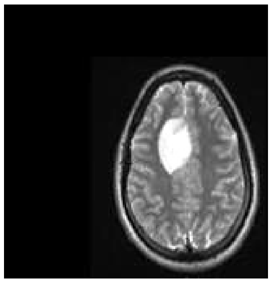
Figure 5: (a) Image 3

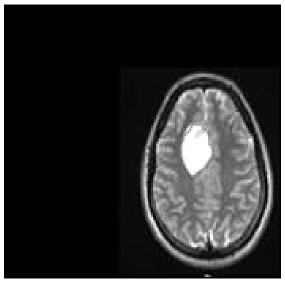
Figure 5: (b) Image 3

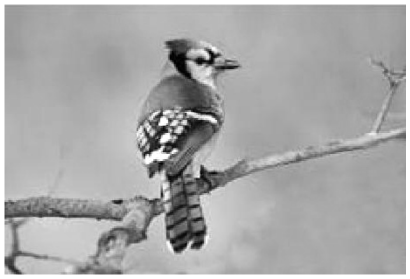
Figure 6: (a) Image 4

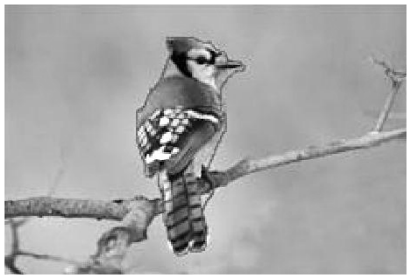
Figure 6: (b) Image 4

Figure 7: (a) Image 5

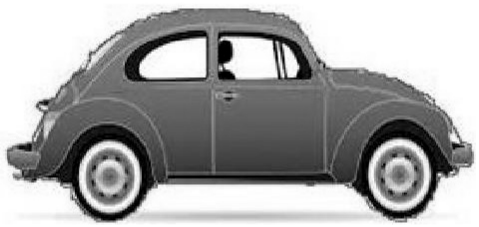
Figure 7: (b) Image 5

Figure 8: (a) Image 6

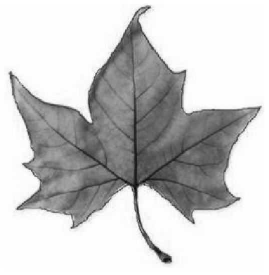
Figure 8: (b) Image 6

Execution time comparison of table for the above shown images is summarized in the following two tables. Table 6(a) depicts the time needed for execution for both the serial and parallel version of the GVF Snake method in quad core Pentium architecture and Table 6(b) shows the execution details in 8 core system in Windows Operating System.

Figure 6: (a) Execution time comparison in 4 core system

Name	*Image Size*	*Matlab (in Sec)*	*Java (in Sec)*	*C (in Sec)*	*OpenMP (in Sec)*
Image 1	527 KB	8.23	9.65	10.21	4.19
Image 2	590 KB	8.58	9.39	9.87	4.54
Image 3	600 KB	10.43	10.68	10.95	5.11
Image 4	820 KB	11.01	11.54	12.08	6.07
Image 5	4 MB	20.65	21.91	22.90	10.28
Image 6	6 MB	40.40	41.26	43.82	20.53

Figure 6: (b) Execution time comparison in 8 core system

Image	*Image Size*	*Matlab (in Sec)*	*Java (in Sec)*	*C (in Sec)*	*OpenMP (in Sec)*
Image 1	527 KB	4.98	5.25	5.80	1.27
Image 2	590 KB	4.65	4.79	4.90	1.45
Image 3	600 KB	5.13	5.02	5.34	1.17
Image 4	820 KB	5.93	6.02	6.08	2.01
Image 5	4 MB	10.56	12.19	13.09	5.98
Image 6	6 MB	21.39	25.32	24.12	10.35

8. CONCLUSION AND FUTURE SCOPE

From the view point of high performance computing, the processing complication of calculating Gradient Vector Field with Snake deformation to segment the convergent region of interest can be reduced significantly by adopting OpenMP parallel directives to utilize parallel processing in multicore architecture. The large number of processing loops that are quite iterative in nature can be parallelized with loop level and data level parallelism with suitable

OpenMP shared constructs. Resultant aspects of this work reveals that shared memory parallel constructs like OpenMP can be very well utilized to attain maximum processing speed and efficiency in the segmentation of medical images. Their implementation to the existing system is quite simple and easy to adapt with better cost effectiveness. The execution time difference for both sequential and parallel versions can be better compared so as to compare the computational capacity of multicore CPUs. Research shows that best speed is achieved when process to thread communication is done in an atomic and mutually exclusive manner. Future research will be focussing on extending the parallelized implementation of various image segmentation algorithms to improve the processing speed, efficiency and accuracy of medical image segmentation applications as the need for effective parallel software development continues to grow in importance.

REFERENCES

[1] Rigo Alvarado, Juan J Tapia, Julio C Rolon, Medical Image Segmentation with Deformable Models on Graphics Processing Units, Journal of Super Computing, Springer, December 2013.

[2] Rhonda D Philips, Layne T Watson, Randolph H Wyne (2006), Hybrid image classification and parameter selection using a shared memory parallel algorithm, Elsevier.

[3] Mahmoud MKA, Al-Jumaily A (2011) segmentation of skin cancer images based on GVF snake, IEEE International Conference on mechatronics and automation, China pp. 216-220.

[4] Schellmann M, Gorlatch S, Meilander D, Kosters T, Schafers K, Wubbeling F, Burger M (2011), Parallel medical image reconstruction from graphics processing units to grids, Journal of Supercomputing 57:151-160.

[5] Pallippuram VK, Bhuiyan M, Smith M C(2012) A comparative study on GPU programming models and architectures using neural networks, Journal of Supercomputing 61:673-718.

[6] Zuoyong Zheng, Ruixia Zhang (2012) A fast GVF snake algorithm on the GPU, Research Journal of Applied Sciences, Engineering and Technology 4(24):5565-5571.

[7] Seyong Lee, Rudolf Eigenmann (2012) OpenMPC: Extended OpenMP for efficient programming and tuning on GPUs, International Journal of Computational Science and Engineering, Vol 7, No:1.

[8] Using OpenMP – Portable Shared Memory Parallel Programming, Barabara Chapman, Gabriele Jost, Ruudo Van Der Pas.

[9] He Z, Kuester F (2006), "GPU based Active Contour Segmentation using Gradient Vector Flow", Advances in Visual Computing Second International Symposium, ISVC 2006Lake Tahoe, USA.

[10] Zheng Z, Zhang R (2011), "A GPU accelerated GVF Snake Algorithm", Proceedings of 2011 workshop on digital media and Digital Content Management, DMDCM, Hangzhou, China.

[11] Perrot G, Domas S, Couturier R, Bertaux N (2011), "GPU Implementation of region based algorithm on large image segmentation", In 11th IEEE International Conference on Computer and Information Technology, Belfort, France.

[12] Li Wang, Chunming Li, Quansen Sun, Deshen Xia, Chiu-Yen Kao (2009), Active Contours driven by local and global intensity fitting energy with application to brain MR image segmentation, Science Direct, Elsevier, Vol. 33, Issue 7.

[13] "Automatic detection of lung cancer nodules by employing intelligent fuzzy cmeans and support vector machine ", Biomedical Research,August 2016 Impact Factor: 0.226 (SCI, Scopus indexed).

[14] "Cognitive Computational Semantic for high resolution image interpretation using artificial neural network", Biomedical Research, August 2016 Impact Factor: 0.226(SCI, Scopus indexed).

[15] Jinliang Wan, Yanhui Liu (2013), Hybrid MPI-OpenMP Parallelization of image reconstruction, Journal of Software, Vol. 8, No. 3.

[16] W. Kim and C. Kim, "Active contours driven by the salient edge energy model," IEEE Transactions on Image Processing, Vol. 22, No. 4, pp. 1667-1673, 2013.

[17] J. Zhao, G. Liang, Z. Yuan, and D. Zhang, "A new method of breakpoint connection using curve features for contour vectorization," Electronics and Electrical Engineering, Vol. 18, No. 9, pp. 79-82, 2012.

[18] Jianhui Zhao, Bingyu Chen, Mingui Sun, Wenyan Jia, and Zhiyong Yuan, Improved Algorithm for Gradient Vector Flow Based Active Contour Model Using Global and Local Information, The Scientific World Journal Volume 2013 (2013), Article ID 479675.