An Improved Image Segmentation Algorithm Based on GPU Parallel Computing

Haiyang Li
College of Mathematics & Computer Science, Mianyang Normal University, Sichuan Province, P.R.China
Email: lhy1301@126.com

Zhaofeng Yang and Hongzhou He
School of Software Engineering, Pingdingshan University, Henan Province, P.R.China
College of Mathematics & Computer Science, Mianyang Normal University, Sichuan Province, P.R.China
Email: pdsncyiyang@163.com, zmoonmoonlhm@aliyun.com

***Abstract*—In the process of image segmentation, the classic Fuzzy C-Means (FCM) algorithm is time-consuming and depends heavily on initialization center. Based on Graphic Processing Unit (GPU), this paper proposes a novel FCM algorithm by improving the computational formulas of membership degree and the update criterion of cluster centers. Our algorithm can initialize cluster centers purposefully and further optimize them according to the analysis on the thread model of the graphic hardware. The compared experimental results with the classic FCM algorithm show that our algorithm has obvious superiority in improving image segmentation quality and efficiency.**

***Index Terms*—Image Segmentation, Graphic Processing Unit, parallel computing, Fuzzy C-Means**

I. Introduction

Image segmentation is to distinguish the different regions of special significance in the image. Usually, these regions are mutually disjoint and their characteristics, such as grain, texture, color etc are very similar. There are various image segmentation algorithms in the literature. They belong to different categories. Cheriet [1] presented a general recursive approach for image segmentation by extending Otsu's method, which succeeded in the scope of document images. Lalonde [2] presented a template matching approach for image segmentation used in low-resolution color medical images. Seunghwan [3] presented a region growing algorithm for image segmentation used in image restoration. Yang [4] proposed a fine edge-preserving de-interlacing algorithm used in image scanning. LI [6] proposed novel PCNN (Pulse Coupled Neural Network) parameters automatic decision algorithm.

The clustering is a frequently-used digital image segmentation technique and the FCM algorithm is one of the most widely used methods among others. Dunn [7] advocated FCM algorithm for the first time in 1973. Bezdek [8] spread it to the cluster analysis in 1981. The basic idea of FCM is to obtain the maximum value by iteratively optimizing objective function, and thus resulting in optimal clustering result [9]. However, FCM algorithm is sensitive to initialization center and very time-consuming in processing high-dimension data [10]. In order to overcome the first drawback, SUN [11] used genetic evolution method and Zhang [12] improved the objective function. LI [13] proposed a fast FCM by improving its membership function. Chen [14] proposed an efficient FCM based on the quad-tree. Yang [15] proposed an adaptive FCM clustering algorithm, which can solve the problem of local optimum. Liu [16] proposed a Gaussian kernel-based fuzzy c-means algorithm with spatial information, which has the better performance.

These algorithms can successfully improve the efficiency of FCM algorithm segmentation. However, FCM algorithm is still difficult to meet the needs of many applications, especially in real-time computationally intensive image segmentation if they run serially on CPU.

GPU is a parallel vector processor. It shows excellent performance in operating parallel data. The optimization of image segmentation algorithm On GPU is becoming a research interest in recent years. Based on GPU, Reza [17] and Zechner [18] have a great success in enhancing the segmentation speed of K-means algorithm. However, there is little research on enhancing the segmentation speed of FCM algorithm by GPU parallel computing.

Based on NVIDIA Compute Unified Device Architecture (CUDA), we propose an improved FCM clustering algorithm. Inspired by the ideas in [12] and [13], we improve the calculation formulas of membership degree and the update criterion of cluster centers. We choose the GPU as computing units to design and realize our algorithm. The comparing results with the classic FCM clustering algorithm indicate that our algorithm has better visual effect and segmentation efficiency.

This work was supported by Science & Technology Department of Sichuan Province and Mianyang Normal University, China under Grant No. 2012JYZ013 and No. 2013A12.
Corresponding author: Zhaofeng Yang

II. FCM Algorithm and Its Improvement

A. FCM Algorithm

In the FCM algorithm, each sample can belong simultaneously to more than one class because of different degree of membership. Each class is a fuzzy subset of the sample set. The classification matrix corresponding to some classification result is called fuzzy classification matrix.

Consider an image which constitutes a dataset X ($X = \{X_1, X_{2,} \dots, X_n\}$) that has to be clustered into c centers. Let u_{ik} be the $i-th$ degree of membership of the $k-th$ pixel. Classification results can be expressed by a fuzzy membership matrix like $U = (u_{ik})$ .Let $J_m(U,V)$ be the objective function of membership matrix U and cluster center matrix V ($V = \{V_1, V_2, \cdots, V_c\}$). Minimizing the value of function $J_m(U,V)$ can achieve the FCM clustering. The minimization process is calculated as follow:

$$J_m(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m d_{ik}^2(x_k, v_i) \qquad (1)$$

Where $2 \le c \le n$ and $m \in [1, +\infty)$. The m is the fuzzy weighted index. It controls the data clustering process degree of fuzzy. If m=1, fuzzy clustering degenerate into hard C-means clustering. Recommended value of m in [19] is $m \in [1.5, 2.5]$. Usually, m = 2.

The expression $d_{ik}^2(x_k, v_i)$ is the distance of the $k-th$ pixel to the $i-th$ class center. It is calculated as follow:

$$d_{ik}^2(x_k, v_i) = \|x_k, v_i\|_A^2 = (x_k - v_i)^T A(x_k - v_i) \qquad (2)$$

Here A is a positive definite matrix. When A is a unit matrix (A=I), the distance is Euclidean distance. FCM algorithm could be realized finally by iterating and optimizing the objective function repeatedly. Its perform steps are shown as follows:

Step1 Initialize the cluster center

$$V = \{V_1, V_{2,} \cdots, V_n\}$$

Step2 Calculate the membership matrix

$$u_{ik} = \frac{1}{\sum_{j=1}^{c}\left(d_{ik}(x_k, v_i) \big/ d_{jk}(x_k, v_j)\right)^{2/(m-1)}}, k = 1, 2, \cdots, n \qquad (3)$$

Step3 Update cluster centers

$$v_i = \frac{\sum_{k=1}^{c}(u_{ik})^m x_k}{\sum_{k=1}^{c}(u_{ik})^m}, i = 1, 2, \cdots, c \qquad (4)$$

Step4 Repeat steps 2 and 3 until the formula (4) converge.

B. Improved FCM Algorithm

In the FCM algorithm, the adjacent two iterations can get two cluster centers. If the distance between these two cluster centers is less than some threshold, we can determine that the FCM algorithm has converged. The number of the objective function can be reduced by each iterative operation of the algorithm. However, objective function $J_m(U,V)$ may have multiple extreme points. If the initial cluster centers are selected in the vicinity of a local minimum, it may cause the algorithm to converge to a local minimum. It means that the algorithm is sensitive to initial value and its classification accuracy rate is decreased.

To solve these problems, we propose an improved FCM algorithm. Let $X = \{X_1, X_{2,} \dots, X_n\}$ be a set of classified samples. Minimum distance threshold between classes is R. Cluster center initialization steps are shown as follows:

Step1 Calculate the distance between any two samples to generate the distance matrix D. Put the nearest two samples into a category and let the midpoint of the two samples as the cluster center of the first class.

Step2 Using the distance matrix D, find all samples which distance to the first cluster center greater than R. In these samples, classify the nearest two samples into a category and let the midpoint of the two samples as second category cluster center.

Step3 Similarly, like step1 and step2, find two nearest samples in remain samples. Classify the two samples into a category and let the midpoint of the two samples as new category cluster center.

Step4 Repeat step 3 until the category c is found.

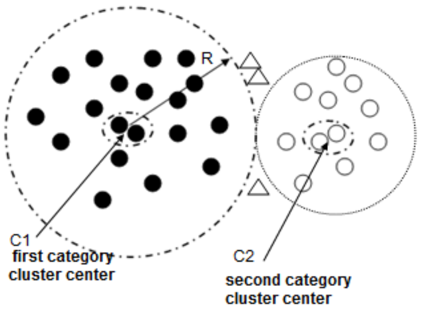

Figure 1. Initialization of cluster centers

In accordance with the requirements of steps 1 and 2, the samples should be divided into two categories. The selection process of initial cluster center is shown as Fig. 1. As can be seen from Fig.1, initialization of cluster centers is executed in multiple areas. This method can make the clustering process avoid local convergence; thereby reduce the dependence of FCM algorithm on initialize cluster centers.

As Fig.1 shown, when the cluster center of C1 is selected, we use the method of searching distance matrix D to determine whether the distance of other samples to the first cluster center is greater than a threshold value R. It need not calculate the distance of cluster center to the other sample points, thus avoiding a lot of Euclidean distance calculation and simplifying the process of cluster centers initialization. However, this process will result in some deviation, i.e. there are some samples not in any class. For example, the triangle samples shown in Fig.1 are neither in C1, nor in C2. To reduce this deviation, we have to reduce the threshold value of R and modify the objective function.

The FCM clustering process finds the optimal classification based on the objective function. As previously mentioned, the objective function is the weighted square distance of all points to the corresponding cluster centers. It does not consider the mutual influence between cluster centers. This paper presents an objective function. It considers not only the distance of data points to the cluster centers, but also the distance between the cluster centers, which ensure the tightness of data points in the same class and the separation of the data points in different classes, consequently avoiding the local optimal of the clustering result. The new objective function is:

$$J_m(U,V,\eta)=\sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m d_{ik}^2(x_k,v_i)-\frac{1}{c(c-1)}\sum_{i=1}^{c}\sum_{j}^{c}(\eta_{ij})^m d_{jk}^2(v_i-v_j) \quad (5)$$

$$\eta_{ij}=\frac{\min\{d_{ik}(v_i,v_k),d_{jm}(v_j-v_m)\}}{\max\{d_{mn}(v_m,v_n)\}},m,n\in c \quad (6)$$

The improvement of objective function is embodied in the expression $\frac{1}{c(c-1)}\sum_{i=1}^{c}\sum_{j}^{c}(\eta_{ij})^m d_{jk}^2(v_i-v_j)$. It can be seen as a punitive function. Equal (5) describes the mutual influence between the clustering centers in the clustering process. If $\eta_{ij}=0$, equal (5) will degenerate to equal (1). The punitive function minimizes the weighted square distance of each data point to the cluster center and maximizes the weighted square distance between the cluster centers simultaneously, consequently alleviating significantly the deviation mentioned above.

Based on Lagrange multipliers, let $\frac{\partial J_m(U,V,\eta)}{\partial J_m(U,V)}=0$, the updating formula of membership can be deduced as follows:

$$u_{ik}=\frac{1}{\sum_{j=1}^{c}\left(d_{ik}(x_k,v_i)\Big/d_{jk}(x_k,v_j)\right)^{2/(m-1)}},k=1,2,\cdots,n \quad (7)$$

$$v_i=\frac{\sum_{k=1}^{n}(u_{ik})^m x_k-\frac{1}{c(c-1)}\sum_{k=1}^{c}(\eta_{ij})^m v_k}{\sum_{k=1}^{n}(u_{ik})^m-\frac{1}{c(c-1)}\sum_{k=1}^{c}(\eta_{ij})^m},i=1,2,\cdots,c \quad (8)$$

By above improvement, FCM algorithm can purposefully initialize cluster centers and overcome its side effect. Thus, FCM algorithm can be less or no longer sensitive to the initialization centers. However, the FCM is an iterative algorithm. It must calculate u_{ik} and v_i repeatedly until converge. In the image segmentation，the data amount of image sample is very large. For example, segmenting an image of 655356 (256 * 256) pixel points will take a large amount of time in performing the FCM algorithm.

III. FCM Algorithm Based on GPU

A. CUDA Computing Model

CUDA consists of two parts. One is hardware drivers and the other is software at different levels. It provides a set of interface for GPU programming. The software structure of CUDA is shown in Fig.2.

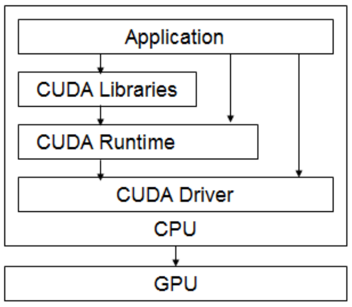

Figure 2. CUDA software structure

In CUDA, GPU can be seen as a computing unit running many threads parallel. If the same function deals repeatedly with different data, it can be called by many different GPU threads. The compiler translates this function into set of instruction executed on GPU. This set of instruction is called objective code or Kernel. Kernel is executed in the form of grid and different grid executes different kernel. Each grid is composed of several thread blocks and each thread block is composed of up to 512 threads. When the programs are initializing, the main program allocates the number of threads and thread blocks, then each thread could locate and execute data by its ID.

B. Parallel Design

FCM clustering algorithm can be divided into three main stages:

- Initialize the cluster center-- select seed.
- Calculate the membership matrix--mark data point.
- Update cluster centers.

The membership degree of pixel is obtained by the distance between the pixel and the clustering center. The processed data are different and independent each other, which can take advantage of the GPU parallel processing. Calculating new cluster center is also suitable for parallel processing on the GPU.

Based on the above analysis, the task division of CPU and GPU is shown in Fig.3.

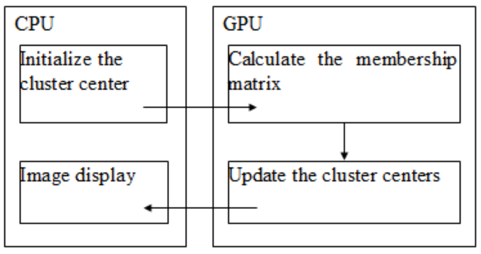
CPU
Initialize the cluster center
Image display
GPU
Calculate the membership matrix
Update the cluster centers

Figure 3. Task division of CPU and GPU

As shown in Fig.3, CUDA realization is divided into two parts. One is to initialize the cluster center and the other is to calculate the membership matrix and update cluster centers. Initializing clustering center is arranged to CPU side due to its small amount of computation.

Calculating the membership matrix and updating cluster centers are arranged to GPU side due to their large amount of computation. A module corresponds to a Kernel. This design can take full advantage of the CUDA platform's characteristics to deal with intensive data and reduce the running time of the application program, consequently improve program performance.

C. Parallel Implementation

According to the parallel design, the perform steps of our algorithm are shown as follows:

Step1 Initialize the cluster centers on CPU. Then, load the data to GPU.

Step2 Set the number of thread blocks using the width of the image matrix.

Step3 Set the thread numbers for each thread block. Usually, the thread number within a block is 64.

Step4 Run the Kernel and calculate pixel membership.

Step5 Set a thread block, its thread number is the number of clusters.

Step6 Run the Kernel and parallel compute the new cluster centers.

Step7 Compare the difference of membership matrix to determine whether it is less than a specified value. If the difference of matrix is greater than or equal to the specified value, turn to step2, or else perform the next step.

Step8 Set the number of thread blocks using the width of the image matrix.

Step9 Set the thread numbers for ach thread block.

Step10 Run the Kernel; classify the pixels to the corresponding category and display.

The steps above are shown as Fig.4.

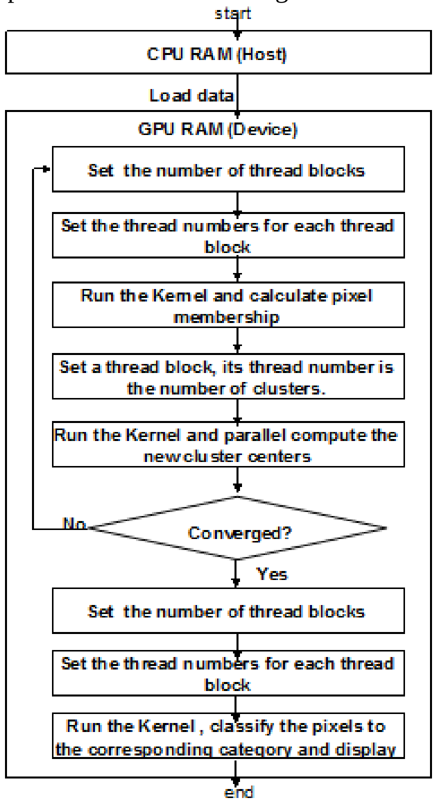
start
CPU RAM (Host)
Load data
GPU RAM (Device)
Set the number of thread blocks
Set the thread numbers for each thread block
Run the Kernel and calculate pixel membership
Set a thread block, its thread number is the number of clusters.
Run the Kernel and parallel compute the new cluster centers
No
Converged?
Yes
Set the number of thread blocks
Set the thread numbers for each thread block
Run the Kernel, classify the pixels to the corresponding category and display
end

Figure 4. Flowcharts of Our algorithm

D. Problem

Typically, the adjacent pixels of the image are associated. This is the basis of image segmentation. However, it will cause great conflicts. As shown in Fig.5, we update the cluster centers using CUDA atomic operations. We can deal with an image including M * N pixels using N threads. Each thread can deal M pixels. It need determine the attributive cluster center of a pixel. However, when multiple pixels are correlated, the adjacent thread will update the same center. This will cause access conflict to the cluster center. The more the number of the cluster center is, the more frequently the conflict is.

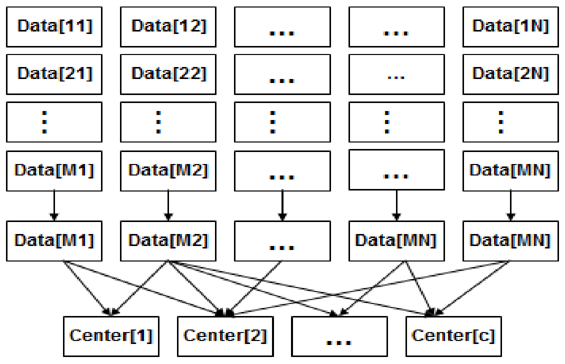

Figure 5. Concurrency conflicts on updating cluster centers

E. Solution

To solve this problem, we allocate an array for each thread using global memory. Each thread will execute an operation of partial update cluster center. The numbers of thread are gradually decreasing in the updating process. As shown Fig.6, updating process finish when number of thread is only one. Because different block can not communicate with each other, it's necessary to reorganize the data to fewer blocks unless only one block remained. The advantage of doing so is that the conflicts described above can be eliminated. Furthermore, it is not necessary to care about the distribution of data.

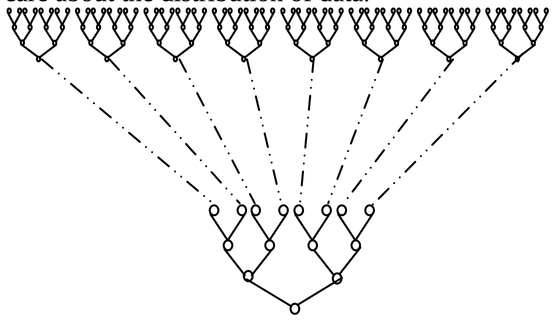

Figure 6. Operating of reduce threads on updating cluster centers

The update operation of cluster centers should be done in the global memory. However, there are hundreds of clock cycles needed to access global memory. In order to avoid unsynchronized global memory access, the shared memory should be used as much as possible. When the shared memory is not enough, the global memory can be used. In this case, operation times to shared memory can be reduced.

IV. Experimental Results

We test our algorithm on numerous outdoor and indoor images. Four of them are chosen to visualize the performance of our algorithm. These images are called Sweet (the size is 101KB; the dimensions are 680 * 660 pixels), Panda (the size is 60KB; the dimensions are 512 * 420 pixels), Bird (the size is 39KB; the dimensions are 256 * 256 pixels), Tiantan (the size is 53KB; the dimensions are 555 * 434 pixels), as shown in Fig.7 (a) to (d) respectively.

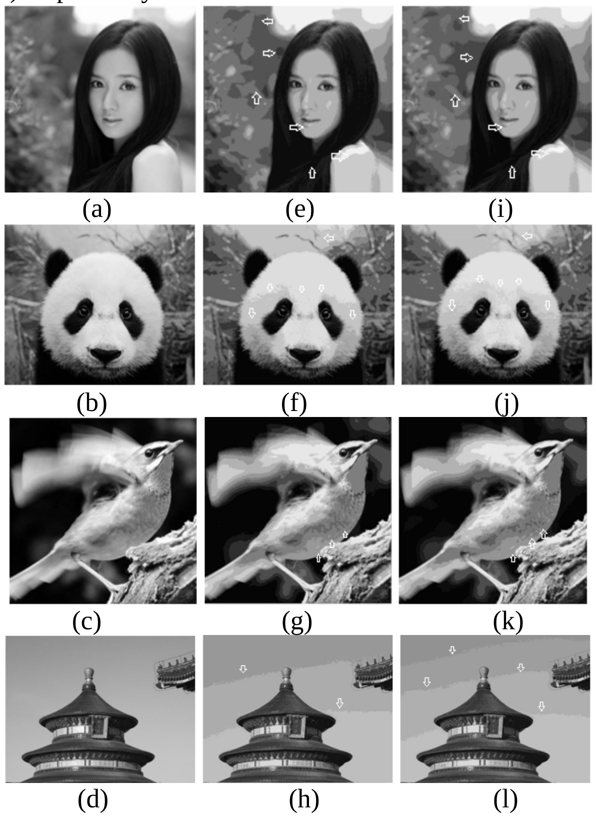

Figure 7. Experimental results of two algorithms

In order to further prove the performance of our algorithm, the classic FCM algorithm is used as a comparison. They have the same experimental environment and parameters. Operating system is Microsoft Windows 7; Memory capacity is 8GB; CPU is Intel Core 2 Duo P8600@2.40GHz;GPU is NVIDIA GTX260 with 192 Stream Processors inside. The fuzzy weighted index m is set to 2，the convergence precision is set to 0.00001, the positive definite matrix A=I and the number of clusters is set to 8. For our algorithm, the threshold R is set to 20. The experimental results of FCM algorithm are shown in Fig.7 (e) to (h) respectively. The experimental results of our algorithm are shown in Fig.7 (i) to (l) respectively. Significant changes on the images are highlighted with arrows.

The classic FCM failed in maintaining the size and the shape of the girl's hair tips and the background leave while our algorithm succeeded (as shown in Fig.7 (e) and (i)). At the girl's lips and shoulder, our algorithm has more detailed segmentation effects than the classic FCM. For the image Panda, our algorithm has more detailed segmentation effects than the classic FCM at the head and cheeks. For the image Bird, our algorithm has more detailed segmentation effects than the classic FCM at the chest feathers, although the classic not so obvious. For the image Tiantan, our algorithm has more detailed segmentation effects than FCM at the background sky.

The execute time of two algorithms are tabulated in Table I.

For all four images, it takes much less time to performing our clustering algorithm than the classic FCM algorithm. For each image, the processing speed of our algorithm is at least 10 times faster than the classic FCM algorithm. We can roughly get similar result when testing the other images.

TABLE I.
EXECUTE TIME OF TWO ALGORITHMS(IN SECONDS)

Images	Algorithms	
	FCM algorithm	Our algorithm
Sweet (s)	84.99	7.85
Panda (s)	68.94	6.23
Bird (s)	32.67	3.21
Tiantan (s)	38.98	3.76

V. CONCLUSION

As an intuitive and easily realized clustering algorithm used in image segmentation, FCM can obtain good segmentation results. However, FCM algorithm depends heavily on initialization center. It is also very time-consuming. These drawbacks limit its application in many fields. Our study proposes an improved FCM algorithm based on GPU parallel computing. In our study, we improve the formulas of classic FCM algorithm by modifying the calculation formulas of membership degree and the update criterion of cluster centers. We further optimize this algorithm based on the parallel computing model and the parallel analysis of the graphic hardware. It is experimentally shown that our FCM algorithm can not only obtain more detailed segmentation effects, but also decrease a large amount of segmentation time.

ACKNOWLEDGMENT

This research was sponsored by Science & Technology Department of Sichuan Province and Mianyang Normal University, China under Grant No. 2012JYZ013 and No. 2013A12.

REFERENCES

[1] M. Cheriet, J. N. Said, and C. Y. Suen, "A recursive thresholding technique for image segmentation," IEEE Transactions on Image Processing, vol. 7,no. 6, pp. 918-921, 1998.

[2] M. Lalonde, M. Beaulieu, and L. Gagnon, "Fast and robust optic disc detection using pyramidal decomposition and Hausdorff-based template matching," IEEE Transactions on Medical Imaging, vol. 21, no. 11, pp.1193-1200, 2001.

[3] Y. Seunghwan, and P. Rae-Hong, "Red-eye detection and correction using inpainting in digital photographs," IEEE