IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 10, 2019, accepted March 23, 2019, date of publication March 26, 2019, date of current version April 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2907573

A Spark-Based Parallel Fuzzy *c*-Means Segmentation Algorithm for Agricultural Image Big Data

BIN LIU[1,2,3], SONGRUI HE[1], DONGJIAN HE[2,3,4], YIN ZHANG[5,6], (Senior Member, IEEE), AND MOHSEN GUIZANI[7], (Fellow, IEEE)

[1]College of Information Engineering, Northwest A&F University, Xianyang 712100, China
[2]Key Laboratory of Agricultural Internet of Things, Ministry of Agriculture and Rural Affairs, Northwest A&F University, Xianyang 712100, China
[3]Shaanxi Key Laboratory of Agricultural Information Perception and Intelligent Service, Northwest A&F University, Xianyang 712100, China
[4]College of Mechanical and Electronic Engineering, Northwest A&F University, Xianyang 712100, China
[5]School of Information and Safety Engineering, Zhongnan University of Economics and Law, Wuhan 430073, China
[6]Anhui Provincial Key Laboratory of Network and Information Security, Wuhu 241002, China
[7]Electrical and Computer Engineering Department, University of Idaho, Moscow, ID 83844, USA

Corresponding author: Dongjian He (hdj168@nwsuaf.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602388, in part by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2017JM6059, in part by the China Postdoctoral Science Foundation under Grant 2017M613216, in part by the Postdoctoral Science Foundation of Shaanxi Province of China under Grant 2016BSHEDZZ121, in part by the Natural Science Foundation of Hubei Province of China under Grant 2017CFB592, and in part by the Fundamental Research Funds for the Central Universities under Grant 2452016081 and Grant 2452015194.

ABSTRACT With the explosive growth of image big data in the agriculture field, image segmentation algorithms are confronted with unprecedented challenges. As one of the most important images segmentation technologies, the fuzzy *c*-means (FCMs) algorithm has been widely used in the field of agricultural image segmentation as it provides simple computation and high-quality segmentation. However, due to its large amount of computation, the sequential FCM algorithm is too slow to finish the segmentation task within an acceptable time. This paper proposes a parallel FCM segmentation algorithm based on the distributed memory computing platform Apache Spark for agricultural image big data. The input image is first converted from the RGB color space to the lab color space and generates point cloud data. Then, point cloud data are partitioned and stored in different computing nodes, in which the membership degrees of pixel points to different cluster centers are calculated and the cluster centers are updated iteratively in a data-parallel form until the stopping condition is satisfied. Finally, point cloud data are restored after clustering for reconstructing the segmented image. On the Spark platform, the performance of the parallel FCMs algorithm is evaluated and reaches an average speedup of 12.54 on ten computing nodes. The experimental results show that the Spark-based parallel FCMs algorithm can obtain a significant increase in speedup, and the agricultural image testing set delivers a better performance improvement of 128% than the Hadoop-based approach. This paper indicates that the Spark-based parallel FCM algorithm provides faster speed of segmentation for agricultural image big data and has better scale-up and size-up rates.

INDEX TERMS Fuzzy C-means, image segmentation, image big data, Apache Spark, parallel algorithm.

I. INTRODUCTION

With the development of mobile devices, sensor networks, Internet of things (IoT), remote sensing, cloud computing and big data, attention is increasingly paid to big data, and the agricultural image big data era has arrived [1]–[5]. As the basis of agricultural image analysis and image understanding, agricultural image segmentation algorithms play a crucial role in agricultural product detection, crop disease and insect pest identification [6], [7], crop nutrient deficiency analysis [8], [9], seed quality inspection [10], precision spraying [11], fruit picking [12], [13], etc. However, due to the high volume, complexity and fast-changing characteristics of agricultural image big data, traditional image segmentation algorithms for

The associate editor coordinating the review of this manuscript and approving it for publication was Zhanyu Ma.

small image data are not applicable [2], [14]–[18]. Therefore, developing image segmentation algorithms for image big data is a research focus.

Image segmentation is the process of partitioning an image into a number of specific, unique areas and extracting the objects of interest in the image by computer [19]. Clustering algorithms, such as K-means [21] and FCM [22], are the most important image segmentation techniques used to group the image data into clusters based on similarities [20]. The FCM algorithm is the most widely used algorithm in clustering [22], which groups similar features of an image without a prior knowledge of data elements. Furthermore, because agricultural images have more complicated backgrounds and pixel ambiguity in a different category, FCM algorithm, as a type of fuzzy clustering analysis, associates each data element with a membership degree to the cluster to which it belongs, making clustering results more flexible and accurate. Although recent researches have proposed novel FCM algorithms for improving the computational efficiency [23], the cluster process is still a computation-intensive task, where the membership degree must be computed on each pixel point. To overcome the challenge, the FCM algorithm needs to be executed in parallel for quickly obtaining the results of the clustering. In recent years, researchers have implemented a variety of parallel FCM algorithms on multicores [24], [25] and graphic processing unit (GPU) platforms [26]–[28] and obtained certain speedups. However, the existing approaches with more complex parallel programming models show problems of higher communication overhead and lower scalability. With the rapid development of computer architecture, many iterative algorithms are successfully ported to the Apache Hadoop [29]–[31], which is a distributed computing platform for processing big data. But Hadoop reads and writes data from the HDFS for each iteration, which consumes considerable time, and all iterative algorithms didn't obtain a significant increase in speedup. In recent years, Apache Spark is a specially designed distributed in-memory computing platform and caches frequently used data and intermediate results in the distributed memory for iterative computing, eventually, some iterative algorithms obtain significant performance improvement [49]. However, there are almost no efficient and parallel FCM segmentation algorithms for agricultural image big data on Apache Spark. Therefore, designing and implementing a Spark-based parallel FCM segmentation algorithm for agricultural image big data becomes more urgent and very essential.

In this paper, a Spark-based parallel FCM segmentation algorithm for agricultural image big data is proposed. The main contributions of this paper are summarized as follows:

- The Spark-based parallel fuzzy C-means algorithm is first employed to segment agricultural image. In order to solve the storage problem of agricultural image big data, the agricultural image big data are firstly partitioned into point cloud data on Spark and stored in different computing nodes. Then, the FCM algorithm with a flexible fuzzy partitioning feature is selected to segment agricultural big images with complex background. Finally, the parallel fuzzy C-means algorithm is implemented on Spark using Spark MapReduce programming model and achieves a significant increase in speedup.
- A novel parallel fuzzy C-means algorithm based on Apache Spark is proposed. The parallelism of conventional FCM algorithm is firstly identified by the expert in this field. Then, the working mechanism and computation model of Apache Spark are studied, and the parallel FCM algorithm is designed based on Apache Spark. At last, the parallel FCM algorithm is mapped to the Spark platform for improving its performance and speedup.

The parallel fuzzy C-means algorithm is implemented on the Apache Spark platform. Experimental results show that the proposed algorithm can fully exploit the inherent parallelism of the FCM algorithm and accelerate the segmentation speed of agricultural image big data. Finally, the Spark-based parallel FCM algorithm achieves an average speedup of 12.54 on 10 computing nodes and provides a performance improvement of 128% for an agricultural image testing set compared to the Hadoop-based approach.

The remainder of this paper is organized as follows. In Section II, image segmentation, the sequential FCM algorithm and the Apache Spark platform are briefly described. In Section III, based on an image preprocessing technique, the point cloud data are first generated, and then the parallel design and implementation of the FCM algorithm are described in detail. Finally, the image reconstruction technique is presented. Section IV analyzes experimental results. In Section V, related work is introduced and summarized. Finally, this paper is concluded in Section VI.

II. PRELIMINARY

A. IMAGE SEGMENTATION AND THE FCM ALGORITHM

With the development of agricultural modernization processes, image segmentation algorithms have been widely used in agricultural product detection, crop pest diagnosis and identification, crop defect analysis, seed quality inspection, precision spraying and fruit picking, etc., and these algorithms have achieved spectacular research results.

The FCM algorithm is one of the best known unsupervised fuzzy clustering algorithms [32], which is classified as a constrained soft clustering algorithm. The main idea of the FCM algorithm is to classify different clusters by the membership degree of the samples to cluster centers. The FCM algorithm minimizes the objective function J_m shown in Equation (1) by searching for the best cluster centers that achieve good partitioning results. To find the center of a cluster, the sum of the distances from points in different clusters to their centers is used as criteria. The criteria are represented by an

objective function J_m.

$$Jm(U, V, X) = \sum_{i=1}^{c} \sum_{j=1}^{n} uij^m dij^2 \tag{1}$$

$$uij = \left[\sum_{k=1}^{c} \left(\frac{dij}{dkj} \right)^{\frac{2}{m-1}} \right]^{-1} \quad 0 \le uij \le 1, \sum_{i=1}^{c} uij = 1 \tag{2}$$

$$Vi = \frac{\sum_{j=1}^{n} uij^m Xj}{\sum_{j=1}^{n} uij^m} \tag{3}$$

where $X = \{x_1, x_{2,} \cdots, x_n\}$ is the dataset, and $V = \{v_1, v_{2,} \cdots, v_n\}$ is the cluster set. m is the real number that controls clustering fuzziness and denotes the membership degree of each data element to the cluster. In general, m takes the value of 2. u_{ij} is the membership matrix with a size of $c \times n$, where c is the number of clusters and n is the data size. $d_{ij} = \|xj - vi\|^2$ measures the closeness of the data element x_j to the cluster center v_i.

Algorithm 1 FCM Algorithm

Input: Clusters c, the membership degree m and the data size n

Output: Clustering results

1 Set these variables c, m, ε, where ε is the stop threshold;
2 Initialize the center vector V randomly;
3 while $\|V_{new} - V_{old}\| > \varepsilon$ **do**
4 Calculate and update U using Equation (2);
5 Calculate and update V using Equation (3);
6 end while
7 Output clustering results.

Algorithm 1 shows the main steps of the FCM algorithm. The first step is the initialization of these variables regarding the FCM algorithm. The values of c and m are set, and the center values v_{ij} are randomly initialized. Then, the algorithm iteratively updates the values of the membership matrix u_{ij} using Equation (2) and the values of the center vector V using Equation (3). The final step is to check for the stopping condition, which is the difference between the current and the previous membership values. If this value is less than a certain thresholdε, then the computation ends or stops. Otherwise, the computation continues until the stopping condition is satisfied.

B. APACHE SPARK COMPUTING PLATFORM

Apache Spark is an open source cluster computing platform based on memory computing that was developed at Berkeley's AMP Lab [33]. Different from other computing clusters such as Apache Hadoop, Apache Spark adopts a distributed memory model and, in order to achieve high performance, it allows frequently used data and intermediate results to be cached for iterative computation. Apache Spark also encompasses a classic master/workers mode, so that the first step in every Spark application is to construct a reusable thread pool by the working nodes, and all tasks will run in the thread pool.

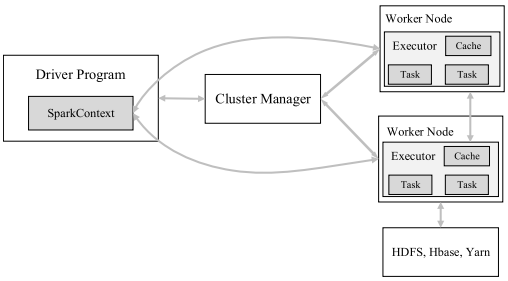


FIGURE 1. Spark's framework.

Figure 1 is a typical Apache Spark cluster with two worker nodes, in which each working node has one or more executor processes. In addition, the running environment that Apache Spark offers is a temporary resource pool composed of these executor processes in all working nodes, and inside each process, there exist some threads that indicate how many tasks (the smallest work units in Spark) can be executed concurrently in one executor. When a program runs on Apache Spark, the threads in all working nodes form a thread pool, and the tasks of the program will run on this thread pool in parallel.

The RDD (resilient distributed dataset) is the most important abstraction in Spark, which is a read-only, partitioned collection of elements that can be operated on in parallel. By splitting records into logical partitions that are distributed across computing nodes in the Spark cluster, the RDD allows Spark to hide data partitions and provide a higher-level programming interface. Partitions are the units of parallelism. Spark supports two kinds of operations to operate these partitions for iterative computation: transformations (e.g., map), which create a new dataset from an existing one, and actions (e.g., reduce), which return a value to the driver program after running a computation on the dataset. The new RDDs are created by transformations, and dependencies are consequently formed between the old RDDs and the new RDDs. Figure 2 shows the two kinds of dependencies.

III. PARALLEL DESIGN AND IMPLEMENTATION OF FCM

A. GENERATING POINT CLOUD DATA

Most of the existing agricultural image big data are saved in the RGB color space. As shown in Fig. 3 [34], the RGB color space is an additive color space, in which red, green and blue lights are added together in various ways to reproduce a broad array of colors. The space sets an RGB value for each pixel of the image to represent the color, and each parameter (RGB) defines the intensity of the color as an integer between 0 and 255.

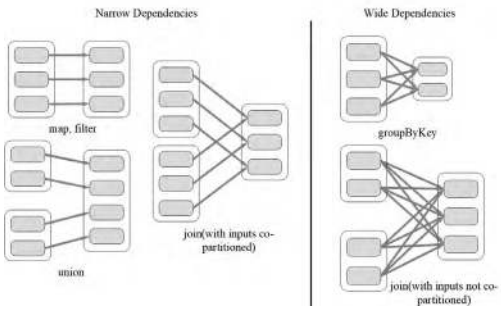


FIGURE 2. Dependencies of RDD.

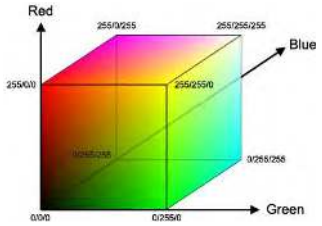


FIGURE 3. RGB color space.

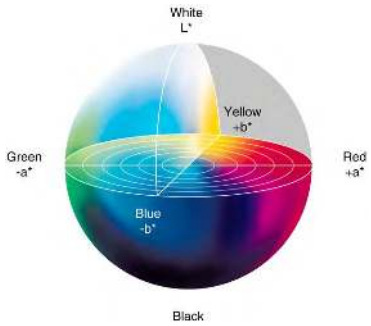


FIGURE 4. Lab color space.

However, the color distribution in the RGB color space is not balanced. The transitional colors between blue and green are too much, and there are no yellow or other colors between green and red. Lab color space is a 3-axis color system with dimension L, a and b. Figure 3 shows the Lab color space [35], where L is the lightness, a is the red/green coordinate, and b is the yellow/blue coordinate. The lab color space is one of the most exact means of representing color, and working with the Lab color space includes all colors in the spectrum. Thus, in this paper, the RGB color space of an agricultural image is first converted to the Lab color space before processing. Equation (4), (5) and (6) are used to implement the conversion.

Because Spark 1.x does not support directly processing the image data, the image data need to be preprocessed into point cloud data, and then these data are stored in the Hadoop distributed file system (HDFS).

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.4124530.357580\ 0.180423 \\ 0.212671\ 0.715160\ 0.072169 \\ 0.019334\ 0.119193\ 0.950227 \end{pmatrix} * \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (4)$$

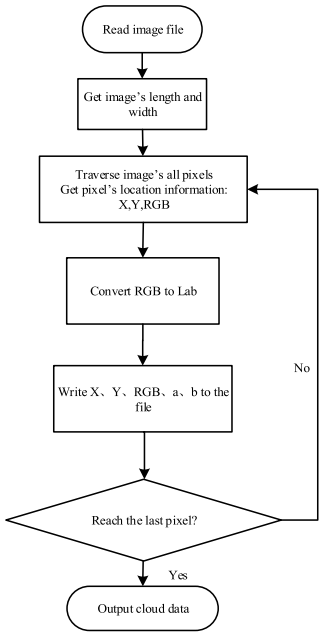


FIGURE 5. The procedure of generating point cloud data.

$$\begin{cases} L = 116f(Y) - 16 \\ a = 500\left[f(\frac{X}{0.982} - f(Y)\right] \\ b = 200\left[f(Y) - f(\frac{Z}{1.83})\right] \end{cases} \quad (5)$$

$$f(x) = \begin{cases} 7.787x + 0.138(\mathrm{x} \leq 0.008856) \\ x^{\frac{1}{3}}(\mathrm{x} > 0.008856) \end{cases} \quad (6)$$

Figure 5 shows the procedure of generating point cloud data. The input image is first read into memory and its length and width are obtained, which are used during image reconstruction. Then, traverse and read all RGB values and their coordinates in the image. Furthermore, using Equation (4), (5) and (6), all RGB values of image data are converted to Lab values to generate strings that are stored in a text file. The values of each line in the text file, including coordinates of each pixel, and their RGB and Lab values, represent a pixel. The text file will be uploaded to the HDFS as point cloud data and stored in a distributed way, which is the input of the parallel fuzzy C-means algorithm based on Apache Spark.

B. IDENTIFYING THE PARALLELISM OF FCM

According to the FCM algorithm, K points are first selected as the initial cluster centers, and each pixel point is assigned to the nearest center, forming K clusters. The membership degree of each pixel point is calculated based on the initial K centers. The process above is iterated until the amount of the variation of the cluster centers is smaller than a specified threshold. In this process, each iteration calculation contains a large number of parallel subprocesses, showing considerable parallelism in the FCM algorithm. Figure 6 shows the parallelism of the FCM algorithm.

Parallelism 1: During the processes of calculating the membership degrees of pixel points to the K cluster centers,

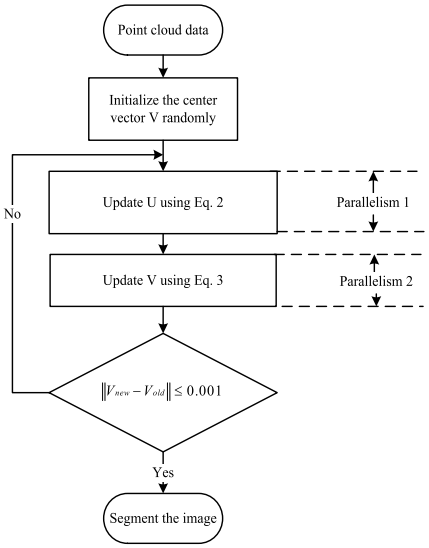


FIGURE 6. The parallelism of FCM.

each pixel point is not associated with others. Therefore, the dataset can be partitioned into different subsets; then, the membership degree in the different subsets between each pixel and the center of each cluster can be calculated in parallel.

Parallelism 2: During the processes of updating new cluster centers using Equation (3), different pixel points in different subsets are independent data; hence, the membership calculation of each pixel point in different subsets to different clustering centers is a good fit for parallel execution.

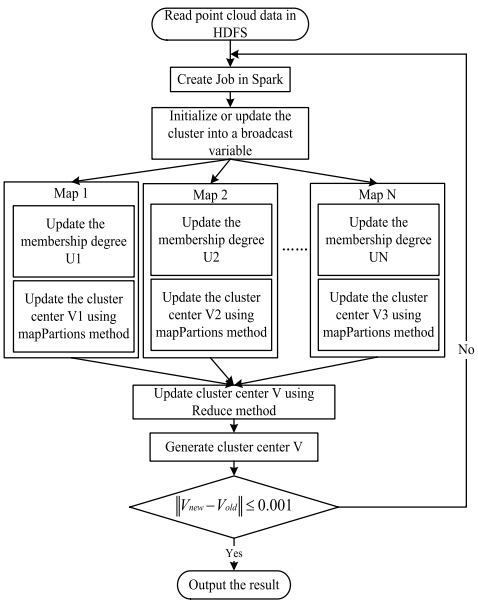


FIGURE 7. The flow of the parallel FCM algorithm.

C. SPARK-BASED PARALLEL FCM ALGORITHM DESIGN AND IMPLEMENTATION

Figure 7 shows the parallel algorithm design of the FCM algorithm based on Spark. The point cloud data are first read into the distributed memory for generating the distributed RDD subsets by the Spark program. Next, the RDD is stored in the distributed memory by executing the Cache operator. After that, the cluster centers are initialized into broadcast variables, which can be shared by different distributed computing nodes, and then the Map operation in the Spark MapReduce programming model is used to compute the membership degree from the pixels to the different cluster centers in parallel. Then, the sum of the membership degrees of all pixels from the same cluster in different computing nodes is computed by the mapPartitions operation in parallel, and finally the sums of the membership degrees of the same cluster center, which come from different computing nodes, are added up using the Reduce operation to generate a new cluster center V. The cluster center is then placed again in a global variable so that it is easily passed to the Map operation for the next iteration. Finally, the clustering criterion is calculated to determine whether the convergence condition is satisfied.

On the basis of designing and analyzing the parallel FCM algorithm based on the Spark platform, this paper implemented a Spark-based parallel FCM algorithm in Scala programming language. The pseudocode is shown in algorithm 2.

Algorithm 2 Parallel FCM Algorithm Based on Apache Spark

Input: Point cloud data
Output: Cluster membership

1 Set the variables k, m,ε;
2 Read point cloud data and generate RDD;
3 while($\|V_{new} - V_{old}\| > \varepsilon$) **do**
4 Initialize randomly or update the cluster center V, and broadcast V to different computing nodes;
5 Compute and update the membership degree U in parallel by the Map operation using Equation (2) across the distributed cluster;
6 Compute and update concurrently the cluster centers V_i, which are part of V, from the *ith* computing nodes by the mapPartitions operation using Equation (3);
7 Collect and summarize the cluster centers *Vi* that come from different nodes using the Reduce operation to generate a new cluster center V;
8 end while
9 Store the cluster information and output the clustered point data.

D. OPTIMAL NUMBER OF CLUSTERS K

As described in Section III C, the FCM algorithm needs to determine the optimum number of cluster centers K before clustering in order to obtain the best image segmentation effect. As shown in Equation (7), the MPC index, proposed by Dave [36] in 1996, is used to compute the optimal

clustering number K.

$$VMPC = 1 - \frac{c}{c-1}\left(1 - \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{n} uij^2\right) \quad (7)$$

where c is the number of clusters, u_{ij} is the membership matrix with a size of $c \times n$, and n is the data size.

The MPC index means that when the sum of the membership degrees of all pixels to their cluster centers reaches the maximum value, the corresponding segmentation effect of the image is at its best, and the maximum value of V_{mpc} is also obtained. At the moment, the corresponding K is the optimal number of clusters. The pseudocode to find optimal K is shown in algorithm 3.

Algorithm 3 Finding the Optimal Number of Clusters

Input: The number of cluster centers K
Output: Optimal number of cluster centers K

1 Initialize $V_{MPC} = 0$, $K = 2$;
2 foreach $i \in (2- > \sqrt{\mathrm{N}})$ **do** // where N is number of pixels
3 Obtain membership degree U by calling Algorithm 2;
4 Calculate V_{MPC} by Equation (9), and set it as a variable *temp*;
5 if $temp > V_{MPC}$ **then**
6 $V_{MPC} = temp$;
7 $K = i$;
8 end if
9 end foreach
10 Output K.

E. IMAGE RECONSTRUCTION

When the parallel FCM image segmentation algorithm is executed, the point cloud data, including coordinate information, the RGB values and class information of pixels are stored in the HDFS. This paper proposed a method to reconstruct the segmented image by point data, as shown in Fig. 8.

The main steps are as follows. First, the width and length of the original image and the number of clusters from the clustered point data are read in order to create K image skeletons. Then, the RGB values and the pixel coordinate positions of the cluster i ($2<i \leq K$) are loaded by line, and the RGB values at the corresponding coordinate positions in the *ith* image skeletons are set to reconstruct the *ith* image. Finally, the above process is repeated until K segmented image files are generated.

IV. EXPERIMENTAL EVALUATION

In this section, the experimental setup is first introduced, and then testing images are provided. Finally, experimental results are analyzed and discussed.

A. EXPERIMENTAL SETUP

Using the Spark MapReduce programming model, the parallel FCM segmentation algorithm for agricultural image big data is implemented in the Scala programming language and runs on the Spark cluster computing platform. The experimental cluster described in this paper is comprised of one master node and twelve worker nodes, interlinked with gigabit Ethernet, and each node in the cluster has the same configuration. As shown in Table 1, each node has an Intel Xeon E7-4820 v4 processor equipped with 8 cores and 16 threads, an 8 G memory and a 600 G disk. In addition, the operating system CentOS 6.5, JDK 1.8.0, Spark 1.6.1, Scala 2.10.4 and Hadoop 2.6.0 are installed on each node to support the execution of the proposed parallel FCM algorithm.

To make full use of Spark's parallelism, the cluster is configured manually. The detailed environment variables are shown in Table 2.

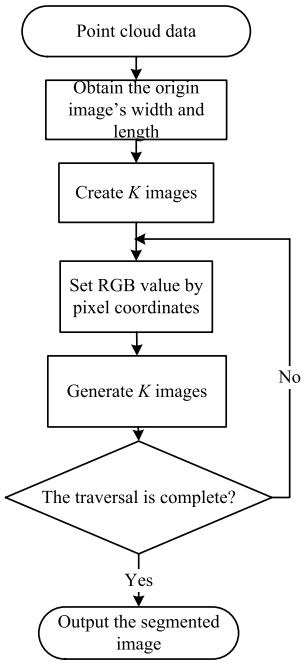


FIGURE 8. The flow chart of the image reconstruction.

TABLE 1. Experimental setup (per node).

Configuration item	Value
CPU	Intel Xeon E7-4820@2.00 GHz
Memory	8 G
Hard Disk	600 G
Operating System	CentOS 6.5
Spark	Spark 1.6.1
JDK	JDK 1.8.0
Scala	Scala 2.10.4
Hadoop	Hadoop 2.6.0

B. THE IMAGE TESTING SET

In this paper, 6 different sizes of pictures in the JPEG format are used as a testing set. These images and their related information are shown in Table 3. These images are all agricultural images captured by Nikon COOLPIX P310.

(a) (b) (c)

FIGURE 9. The segmentation result of the serial algorithm. (a) Original drawing. (b) Background segmentation. (c) Apple segmentation.

(a) (b) (c)

FIGURE 10. The segmentation result of the parallel algorithm on hadoop. (a) Original drawing. (b) Background segmentation. (c) Apple segmentation.

(a) (b) (c)

FIGURE 11. The segmentation result of the parallel algorithm on spark. (a) Original drawing. (b) Background segmentation. (c) Apple segmentation.

TABLE 2. Spark's environment variables.

Property Name	Value	Meaning
spark.executor.instances	1~12	The number of executors in use
spark.executor.cores	4	The number of cores to use on each executor
spark.executor.memory	4 G	Amount of memory to use per executor process
spark.core.max	4*instances	The maximum amount of CPU cores to request for the application from across the cluster

TABLE 3. Description of the testing set.

Name	Number	Description	Resolution	Cloud data
Leaf	1001	Leaves and background	2448*3264	365.2 M
Orange	1002	Oranges and leaves	3456*4608	710.5 M
Apple	1003	Apples and leaves background	7000*5250	1.6 G
Kiwi	1004	Kiwis and leaves background	7250*5438	1.8 G
Golden Delicious	1005	Apples and leaves background	8700*5800	2.3 G
Cotton Fields	1006	Sky and cotton fields	9200*6110	2.5 G

C. EXPERIMENTAL RESULTS AND ANALYSES

In this paper, the testing set was evaluated using a traditional serial approach [37], Hadoop-based approach [30] and our proposed approach. We first focus on the image segmentation effect and the speedup of the proposed parallel FCM algorithm, and then its scaleup and sizeup are also analyzed and discussed.

TABLE 4. MPC index.

Image	MPC			K	Δm_1	Δm_2
	Serial	Hadoop-based	Spark-based			
1001	0.35484142	0.35494144	0.35483172	2	0.00010001	0.00000970
1002	0.52255095	0.52254393	0.52256548	3	0.00000702	0.00001453
1003	0.76082161	0.76081953	0.76079563	2	0.00000208	0.00002598
1004	0.74610347	0.74615671	0.74614644	4	0.00005324	0.00004297
1005	0.66205634	0.66204793	0.66205086	3	0.00000841	0.00000548
1006	0.70359235	0.70349844	0.70349453	3	0.00009391	0.00009782

1) IMAGE SEGMENTATION EFFECT

To validate the segmentation quality of the proposed parallel FCM algorithm, picture 1003 is selected randomly as an example for analysis, and the segmentation results are shown in Fig. 9, 10, 11. Figure 9 shows the result of image segmentation of the serial FCM algorithm. Specifically, Figure 9(a) is the original image, including an apple and a background with apple leaves, branches, soil and sky. Figure 9(b) and (c) are the segmentation results of the serial FCM algorithm. As seen in Fig. 9, apples and their background are completely and precisely segmented out from the image, with a clear contour. Figure 10 and Figure 11 also show the results of image segmentation of Hadoop-based and Spark-based parallel FCM algorithm, respectively. Compared with Fig. 9, intuitively, we can see that there is almost no difference in the segmentation effects in the serial and parallel FCM algorithms.

In addition, in order to quantitatively evaluate the segmentation effects of serial and parallel algorithms, the MPC index is introduced to evaluate the image segmentation effect. Generally speaking, once given k, which is the optimal number of clusters, the execution results of these algorithms should have the same MPC indexes. However, in this paper, K different points are chosen randomly in the FCM initialization, and their MPC indexes will be slightly different. Therefore, MPC values obtained by running these algorithms 5 times are selected as the experimental result. As shown in Table 4, the second, third and fourth columns are MPC indexes of serial, Hadoop-based and Spark-based FCM algorithms, Δm_1 and Δm_2 are, respectively, the difference of MPC indexes between serial and Hadoop-based FCM algorithms and between the serial and Spark-based FCM algorithms in the sixth and seventh columns. From Table 4 we can see thatΔm_1 and Δm_2 are both less than 1×10^{-4}, which indicates that the Spark-based parallel FCM algorithm for agricultural image segmentation has the same segmentation effect as the serial and Hadoop-based FCM approaches.

2) SPEEDUP PERFORMANCE

In this section, the speedup is introduced to evaluate the performance of the Spark-based parallel FCM algorithm, compared with the serial FCM and the Hadoop-based parallel FCM algorithms. The speedup refers to how much faster a parallel algorithm is than a corresponding serial algorithm. The speedup is defined by the following Equation (8):

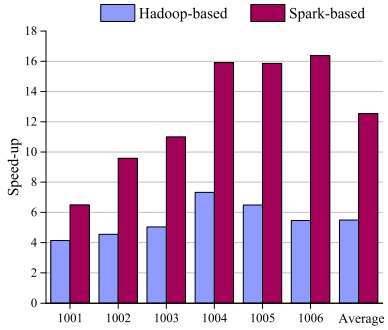


FIGURE 12. The speedup comparison of the hadoop-based and sparkbased fcm algorithms on 10 nodes.

$$\text{Speedup} = \frac{T_1}{T_p} \tag{8}$$

where p is the number of nodes, T_1 is the execution time of the serial algorithm on a single node, and T_p is the execution time of the parallel algorithm.

A higher speedup denotes that less time is consumed by the parallel algorithm. Table 5 shows the speedup of the Spark-based parallel FCM algorithm on different computing nodes. As shown in Table 5, the speedup does not increase linearly with the increased computing nodes. This result is because when the number of nodes increases, the amount of parallel tasks on each node will vary, and the communication cost between nodes changes dynamically, which leads to a fluctuation in speedup. From Table 5, we can clearly see that pictures 1001, 1002 and 1003 obtain the max speedup on 9, 8 and 7 computing nodes, respectively, while pictures 1004, 1005 and 1006 reach the maximum value on 10 computing nodes. As shown on the last line in Table 5, the experimental result also indicates that 10 is the optimal number of nodes, and the Spark-based parallel FCM algorithm achieves the best speed performance. In a similar way, from Table 6, we can reach the same conclusion that the speedup reaches a maximum when the number of nodes is 10 on Hadoop platforms.

To compare the speedup between the Hadoop-based and Spark-based parallel FCM algorithms, testing sets of agricultural images are validated by using these two parallel

TABLE 5. The speedup statistics of spark-based parallel fcm algorithm on different computing nodes.

Testing images	2	3	4	5	6	7	8	9	10	11	12
1001	5.724	6.949	6.621	6.524	3.313	4.569	5.553	**6.919**	6.506	5.425	4.952
1002	7.580	13.100	13.166	14.498	15.251	10.631	**14.318**	14.142	9.579	9.707	9.887
1003	1.820	4.701	10.825	10.710	12.071	**15.759**	14.525	8.731	11.008	8.453	7.249
1004	1.417	2.566	9.853	10.946	13.604	11.067	11.584	12.075	**15.913**	12.811	12.612
1005	2.269	2.742	8.920	11.568	13.670	14.160	15.943	15.353	**15.862**	14.061	13.501
1006	1.671	2.505	4.940	5.626	7.290	8.779	11.393	9.951	**16.387**	14.280	13.722
Average	3.414	5.427	9.054	9.979	10.866	10.827	12.219	11.195	**12.543**	10.790	10.321

TABLE 6. The speedup statistics of hadoop-based parallel fcm algorithm on different computing nodes.

Testing images	2	3	4	5	6	7	8	9	10	11	12
1001	1.65	1.93	2.1	2.89	3.66	3.59	4.03	**4.15**	4.14	3.9	3.16
1002	1.34	2.04	2.45	3.17	3.2	4.11	4.04	4.3	**4.56**	4.51	4.02
1003	1.29	1.77	2.03	2.65	3.06	3.94	4.55	**5.11**	5.04	5.06	4.62
1004	1.3	1.68	2.12	3.37	3.91	4.75	6.13	6.64	**7.33**	7.22	6.94
1005	1.25	1.8	2.54	3.3	3.69	4.77	5.36	6.05	**6.49**	6.45	6.03
1006	1.24	1.75	2.29	2.67	3.11	4.04	4.96	**5.63**	5.46	5.5	5.39
Average	1.35	1.83	2.26	3.0	3.44	4.2	4.84	5.31	**5.5**	5.44	5.03

TABLE 7. The performance improvement.

Testing images	Hadoop-based speedup	Spark-based speedup	Performance improvement
1001	4.14	6.50	57.00%
1002	4.56	9.58	110.09%
1003	5.04	11.00	118.25%
1004	7.33	15.91	117.05%
1005	6.49	15.86	144.38%
1006	5.46	16.38	200.00%
Average	5.5	12.54	128.00%

FCM algorithms on 10 computing nodes. The experimental results presented in Fig. 12 show the overall performance improvements for the Hadoop-based approach and the Spark-based approach. From Fig. 12, we can clearly see that all pictures gain higher speedups in different percentages, and 1005 and 1006 in particular have gained a significant performance improvement. For further analysis, we define the increase rate as:

$$rate\_inc = \frac{Spark\_speedup - Hadoop\_speedup}{Hadoop\_speedup} \times 100\% \quad (9)$$

From the rightmost column in Table 7, we can see that the rates of increase vary from 57.00% to 200.00%, and the average rate of increase can reach 128.00%. Compared with the Hadoop-based FCM algorithm, these experimental results indicate that the Spark-based FCM algorithm can exploit more parallelism and achieve better performance improvement for all pictures. This result is because Spark is an in-memory cluster computing platform developed for iterative algorithms and allows the Spark-based FCM algorithm to directly cache frequently used data and intermediate results in the distributed memory for iterative operation. In contrast, Hadoop reads and writes data from the HDFS for each iteration, which consumes considerable time.

3) SCALEUP

The scaleup is the ability of an m-times larger system to perform an m-times larger job in the same run-time as the original system. The scaleup evaluates the scalability of the algorithm to increase both the system and the dataset size. Equation (10) is shown as follows:

$$\text{Scaleup (data,m)} = \frac{T_1}{T_{mm}} \quad (10)$$

where T_1 is the execution time for processing data on one node, and T_{mm} is the execution time for processing m$*$ data on m computing nodes.

With the increase of computing nodes, the scaleup experiments are performed to verify the ability of the Spark-based FCM algorithm to process larger datasets. In this section, all testing pictures are selected as a dataset, and with the increase of computing nodes, the dataset is replicated and scaled up by factors of 2, 4 and 8 times. For these datasets, the sizes of datasets are executed on 1, 2, 4 and 8 computing nodes.

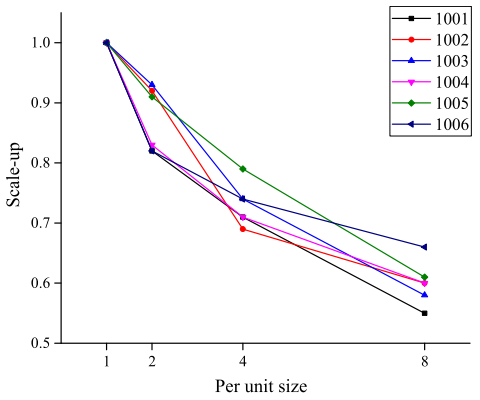


FIGURE 13. The scaleup comparison.

Figure 13 shows the scaleup performance of the datasets. Picture 1006 maintains up to 66% scalability while picture 1002 also maintains a 60% scaleup. Clearly, the Spark-based 1001 FCM algorithm scales very well. However, as the result shows, for all the testing pictures, the scaleup of the Spark-based FCM algorithm is gradually reduced when the number

of computing nodes and the sizes of the datasets increase proportionally. This finding is because when the number of nodes increases, the communication costs between nodes will gradually increase. Consequently, when the data size increases in proportion to the number of computing nodes, the execution time of the parallel algorithm based on Spark will increase.

4) SIZEUP

Sizeup measures how much longer it takes on a given system when the dataset size is m-times larger than the original dataset. It is defined by the following formula.

$$\text{Sizeup (data,m)} = \frac{T_m}{T_1} \tag{11}$$

where T_m is the execution time for processing m$*$ data, and T_1 is the execution time for processing data.

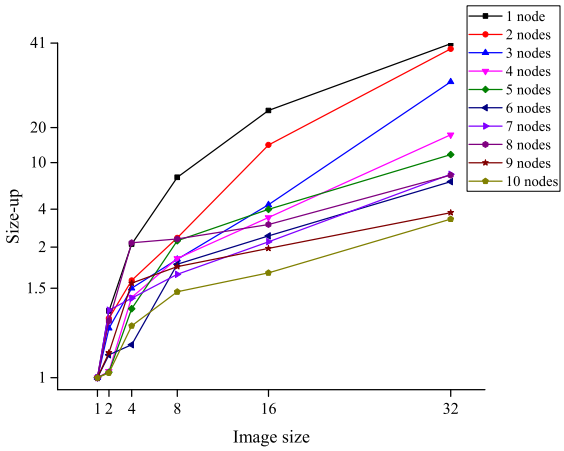


FIGURE 14. The sizeup of different size of photos running the sparkbased fcm algorithm on different numbers of nodes.

The experiment tests different sizes of pictures running the Spark-based FCM algorithm on different numbers of computing nodes, using the same dataset as the experiment of scaleup. As shown in Fig. 14, when the number of nodes is kept constant, sizeup increases proportionally as the size of the dataset becomes larger. When the image size is kept constant, sizeup decreases as the number of nodes increases. Experimental results show that the Spark-based parallel FCM has a good size-up performance. An 32 times larger problem needs about 40 times more time on one node, but only about 3 times more time on 10 nodes. The experiment indicates that FCM algorithm is suitable for the Spark platform, and it can be effectively segmented for agricultural image big data.

V. RELATED WORK

Currently, image segmentation plays an important role in the field of understanding and analysis of image big data such as in medical and agricultural research. In recent years, many algorithms had been proposed for image segmentation [38]–[44]. In [41] and [42], an FCM algorithm is used for MRI image segmentation and local noise detection. In [45], a K-means clustering algorithm and a subtractive clustering algorithm are proposed for image segmentation. In [43], a normalized cuts algorithm is extended for the segmentation of hyperspectral images. In [44], a fuzzy set is used for agricultural image target segmentation. In [46], a discrete wavelet transform is used to distinguish between soil and green parts in agricultural image segmentation.

With the increase in the amount of image big data to be processed, many parallel image segmentation algorithms have been proposed and are implemented on GPU platforms. In [47], Al-Ayyoub *et al.* proposed a GPU-based implementation of FCM algorithms for medical image segmentation. Compared with a CPU-based sequential implementation and a traditional FCM algorithm, GPU-based parallel brFCM is 2.24 times faster than the former, and 23.43 times faster than the latter. In [48], Shehab *et al.* proposed a parallel hybrid CPU-GPU implementation for the FCM algorithm. By executing the membership function and the ''Do segmentation function'' on the GPU card, with the cluster centroids computed and updated on the CPU side, the proposed parallel FCM is 9 times faster than the sequential version. In [26], Ali *et al.* proposed three parallel implementations of the FCM algorithm for MRI image segmentation, in which the first implementation consists mainly of an almost complete data processing hosting at the GPU level through updating the membership matrix, achieving the maximum speedup of 21.16, and the other two implementations are hybrid GPU-CPU methods, achieving the maximum speedup of 5.39 and 13.21, respectively.

With the rapid development of computer architecture, many researchers have implemented parallel algorithms on cluster computing platforms that are specially designed for iterative computing, such as Hadoop and Spark. In [30], Li *et al.* proposed a MapReduce-based fast FCM (MRFFCM) algorithm for large-scale underwater image segmentation, in which a two-layer distribution model is used to group the large-scale images and adopt an iterative MapReduce process to parallelize the FFCM algorithm. Compared to the traditional nonparallel methods, the MRFFCM algorithm can be expected to provide a more efficient segmentation on images with at least 13% improvement. In [49], Zhu *et al.* presented a Spark-based distributed parallel kernel fuzzy C-means (S-KFCM) clustering algorithm for SAR image change detection. Using the Map operation on RDDs, the computation of memberships can be distributed to all nodes in the Spark cluster, and thus the computation of the classification of the change map can be transferred to the Spark cluster. The experimental results show good effectiveness and accelerating performance. Compared to the Hadoop-based KFCM algorithm, the speedup can achieve a maximum of 18.9. In [31], a scalable Fuzzy C-Means clustering method named BigFCM is proposed and designed for the Hadoop distributed data platform. Based on the MapReduce programming model, the proposed algorithm exploits several mechanisms, including an efficient caching design to achieve a several orders of magnitude reduction in execution time. Compared with Apache Mahout K-Means and Fuzzy

K-Means, the BigFCM method shows great scalability while it preserves the quality of clustering. These approaches show some good effects in image segmentation and obtain a certain speedup, but many of these approaches have not been implemented in the field of understanding and analysis for image big data. This paper proposes a parallel Spark-based FCM segmentation algorithm for agricultural image big data and obtains a significant increase in speedup.

VI. CONCLUSION

To solve the problem of image processing and analysis for agricultural image big data, this paper proposes a parallel FCM algorithm based on the Spark distributed computing platform. The point cloud data are first generated from the input image and stored in distributed computing platforms; then, the membership degrees of pixel points to different cluster centers and the cluster centers are calculated and updated in parallel for iterative computing. The segmented image is finally reconstructed based on the clustered point cloud data.

The parallel FCM segmentation algorithm for agricultural image big data is implemented on Apache Spark. The results show that the Spark-based parallel fuzzy C-means algorithm can obtain a significant increase in speedup and reaches an average speedup of 12.54 on 10 computing nodes. From an overall perspective, using an agricultural image testing set, the results are satisfactory, and the parallel FCM algorithm achieves an average of 128% performance improvement.

ACKNOWLEDGMENT

(Bin Liu and Songrui He contributed equally to this work.)

REFERENCES

[1] G. M. Alves and P. E. Cruvinel, "Big data infrastructure for agricultural tomographic images reconstruction," in *Proc. 12th IEEE Int. Conf. Semantic Comput.*, Jan./Feb. 2018, pp. 346–351.

[2] J. Xia *et al.*, "Hyperspectral identification and classification of oilseed rape waterlogging stress levels using parallel computing," *IEEE Access*, vol. 6, pp. 57663–57675, 2018.

[3] J. Santhana Krishnan and P. SivaKumar, "Big data-based image retrieval model using shape adaptive discreet curvelet transformation," in *Proc. Int. Conf. Big data Cloud Comput.*, 2019, pp. 215–227.

[4] S. Ye *et al.*, "Design and implementation of automatic orthorectification system based on GF-1 big data," *Trans. Chin. Soc. Agricult. Eng.*, vol. 33, pp. 266–273, Mar. 2017.

[5] J. Hruška *et al.*, "Machine learning classification methods in hyperspectral data processing for agricultural applications," in *Proc. Int. Conf. Geoinformatics Data Anal. Int. Conf. Softw. Services Eng.*, 2018, pp. 137–141.

[6] N. M. H. Hassan and A. A. Nashat, "New effective techniques for automatic detection and classification of external olive fruits defects based on image processing techniques," in *Multidimensional Systems and Signal Processing*, vol. 32, no. 2. Berlin, Germany: Springer, 2018, pp. 1–19.

[7] B. Liu, Y. Zhang, D. J. He, and Y. Li, "Identification of apple leaf diseases based on deep convolutional neural networks," *Symmetry*, vol. 10, no. 1, pp. 1–16, 2017.

[8] K. Zhang, A. Zhang, and C. Li, "Nutrient deficiency diagnosis method for rape leaves using color histogram on HSV space," *Trans. Chin. Soc. Agricult. Eng.*, vol. 32, no. 19, pp. 179–187, 2016.

[9] P. Balasubramaniam and V. P. Ananthi, "Segmentation of nutrient deficiency in incomplete crop images using intuitionistic fuzzy C-means clustering algorithm," *Nonlinear Dyn.*, vol. 83, nos. 1–2, pp. 849–866, 2016.

[10] X. Wang, W. Yang, and Z. Li, "A fast image segmentation algorithm for detection of pseudo-foreign fibers in lint cotton," *Comput. Electr. Eng.*, vol. 46, pp. 500–510, Aug. 2015.

[11] Y. Xu, Z. Gao, L. Khot, X. Meng, and Q. Zhang, "A real-time weed mapping and precision herbicide spraying system for row crops," *Sensors*, vol. 18, no. 12, p. 4245, 2018.

[12] Y. Yu, Z. Sun, X. Zhao, J. Bian, and X. Hui, "Design and implementation of an automatic peach-harvesting robot system," in *Proc. 10th Int. Conf. Adv. Comput. Intell.*, Mar. 2018, pp. 700–705.

[13] J. Xiong *et al.*, "The recognition of litchi clusters and the calculation of picking point in a nocturnal natural environment," *Biosyst. Eng.*, vol. 166, pp. 44–57, Feb. 2018.

[14] X. Zhang, Y. Yang, and L. Shen, "Spark-SIFT: A spark-based large-scale image feature extract system," in *Proc. 13th Int. Conf. Semantics, Knowl. Grids*, Aug. 2018, pp. 69–76.

[15] K. He, D. Wang, M. Tong, and X. Zhang, "Interactive image segmentation on multiscale appearances," *IEEE Access*, vol. 6, pp. 67732–67741, 2018.

[16] Y. Li and L. Shen, "cC-GAN: A Robust transfer-learning framework for HEp-2 specimen image segmentation," *IEEE Access*, vol. 6, pp. 14048–14058, 2018.

[17] S. Yin, Y. Zhang, and S. Karim, "Large scale remote sensing image segmentation based on fuzzy region competition and gaussian mixture model," *IEEE Access*, vol. 6, pp. 26069–26080, 2018.

[18] L. Zhang *et al.*, "Improving semantic image segmentation with a probabilistic superpixel-based dense conditional random field," *IEEE Access*, vol. 6, pp. 15297–15310, 2018.

[19] K. Suresh and P. S. Rao, "Various image segmentation algorithms: A survey," in *Proc. 2nd Int. Conf. Smart Comput. Informat.*, 2019, pp. 233–239.

[20] L. Huang, H. Chao, and C. Wang, "Multi-view intact space clustering," *Pattern Recognit.*, vol. 86, pp. 344–353, Feb. 2019.

[21] C.-Y. Han, "Improved SLIC imagine segmentation algorithm based on K-means," *Cluster Comput.*, vol. 20, no. 2, pp. 1017–1023, 2017.

[22] J. Arora, K. Khatter, and M. Tushir, "Fuzzy c-means clustering strategies: A review of distance measures," *Proc. 50th Annu. Conv. Comput. Soc. India, Softw. Eng.*, vol. 2019, pp. 153–162.

[23] C. L. Chowdhary and D. P. Acharjya, "Clustering algorithm in possibilistic exponential fuzzy c-mean segmenting medical images," *J. Biomimetics, Biomater. Biomed. Eng.*, vol. 30, pp. 12–23, Jun. 2018.

[24] Y. Li, P. P. Yin, and Y. Zhao, "A FCM-based thread partitioning algorithm for speculative multithreading," *Chin. J. Comput.*, vol. 37, no. 3, pp. 580–592, 2014.

[25] A Ravi, A. Suvarna, A. D'Souza, and G. R. M. Reddy, "Souza, "A parallel fuzzy c means algorithm for brain tumor segmentation on multiple MRI images," in *Proc. Int. Conf. Adv. Comput.*, 2013, pp. 787–794.

[26] N. A. Ali, B. Cherradi, A. El Abbassi, O. Bouattane, and M. Youssfi, "GPU fuzzy c-means algorithm implementations: Performance analysis on medical image segmentation," *Multimedia Tools Appl.*, vol. 77, no. 16, pp. 21221–21243, 2018.

[27] M. Al-Ayyoub, S. AlZu'bi, Y. Jararweh, M. A. Shehab, and B. B. Gupta, "Accelerating 3D medical volume segmentation using GPUs," *Multimedia Tools Appl.*, vol. 77, no. 4, pp. 4939–4958, 2018.

[28] T. Kalaiselvi and P. Sriramakrishnan, "Rapid brain tissue segmentation process by modified FCM algorithm with CUDA enabled GPU machine," *Int. J. Imag. Syst. Technol.*, vol. 28, no. 3, pp. 163–174, 2018.

[29] W. Dai, C. Yu, and Z. Jiang, "An improved hybrid canopy-fuzzy c-means clustering algorithm based on mapreduce model," *J. Comput. Sci. Eng.*, vol. 10, no. 1, pp. 1–8, 2016.

[30] X. Li, J. Song, F. Zhang, X. Ouyang, and S. U. Khan, "MapReduce-based fast fuzzy c-means algorithm for large-scale underwater image segmentation," *Future Gener. Comput. Syst.*, vol. 65, pp. 90–101, Dec. 2016.

[31] N. Ghadiri, M. Ghaffari, and M. A. Nikbakht, "BigFCM: Fast, precise and scalable FCM on hadoop," *Future Gener. Comput. Syst.*, vol. 77, pp. 29–39, Dec. 2017.

[32] L. Liu, C.-F. Li, Y.-M. Lei, J.-J. Zhao, J.-Y. Yin, and X.-K. Sun, "A new fuzzy clustering method with neighborhood distance constraint for volcanic ash cloud," *IEEE Access*, vol. 4, pp. 7005–7013, 2016.

[33] S. Ramirez-Gallego, S. García, J. M. BenÃtez, and F. Herrera, "A distributed evolutionary multivariate discretizer for big data processing on apache spark," *Swarm Evol. Comput.*, vol. 38, pp. 240–250, Feb. 2018.

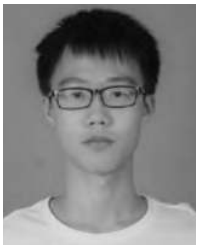
[34] D. Maia and R. Trindade, "Face detection and recognition in color images under MATLAB," *Int. J. Signal Process., Image Process. Pattern Recognit.*, vol. 9, no. 2, pp. 13–24, 2016.

[35] H. A. Eissa, M. T. Ramadan, H. S. Ali, and G. H. Ragab, "Optimizing oil reduction in fried eggplant rings," *J. Appl. Sci. Res.*, vol. 9, no. 6, pp. 3708–3717, 2013.

[36] R. N. Dave, "Validating fuzzy partitions obtained through c-shells clustering," *Pattern Recognit. Lett.*, vol. 17, no. 6, pp. 613–623, May 1996.
[37] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," *J. Cybern.*, vol. 3, no. 3, pp. 32–57, 1973.
[38] Q. Li *et al.*, "Glioma segmentation with a unified algorithm in multimodal MRI images," *IEEE Access*, vol. 6, pp. 9543–9553, 2018.
[39] Z. Liu, C. Cao, S. Ding, Z. Liu, T. Han, and S. Liu, "Towards clinical diagnosis: Automated stroke lesion segmentation on multi-spectral MR image using convolutional neural network," *IEEE Access*, vol. 6, pp. 5706–5716, 2018.
[40] P. Sahare and S. B. Dhok, "Multilingual character segmentation and recognition schemes for Indian document images," *IEEE Access*, vol. 6, pp. 10603–10617, 2018.
[41] J. K. Sing, S. K. Adhikari, and D. K. Basu, "A modified fuzzy C-means algorithm using scale control spatial information for MRI image segmentation in the presence of noise," *J. Chemometrics*, vol. 29, no. 9, pp. 492–505, 2015.
[42] F.-F. Guo, X.-X. Wang, and J. Shen, "Adaptive fuzzy c-means algorithm based on local noise detecting for image segmentation," *IET Image Process.*, vol. 10, no. 4, pp. 272–279, Apr. 2016.
[43] O. Torun and S. E. Yüksel, "Hyperspectral image segmentation using normalized cuts," in *Proc. 24th Signal Process. Commun. Appl. Conf.*, May 2016, pp. 1717–1720.
[44] R. Gao and H. Wu, "Agricultural image target segmentation based on fuzzy set," *Optik*, vol. 126, no. 24, pp. 5320–5324, 2015.
[45] N. Dhanachandra, K. Manglem, and Y. J. Chanu, "Image segmentation using K-means clustering algorithm and subtractive clustering algorithm," in *Procedia Computer Science*, vol. 54. Amsterdam, The Netherlands: Elsevier, 2015, pp. 764–771.
[46] M. Guijarro, I. Riomoros, G. Pajares, and P. Zitinski, "Discrete wavelets transform for improving greenness image segmentation in agricultural images," *Comput. Electron. Agricult.*, vol. 118, pp. 396–407, Oct. 2015.
[47] M. Al-Ayyoub, A. M. Abu-Dalo, Y. Jararweh, M. Jarrah, and M. Al Sa'd, "A GPU-based implementations of the fuzzy C-means algorithms for medical image segmentation," *J. Supercomput.*, vol. 71, no. 8, pp. 3149–3162, 2015.
[48] M. Shehab, M. Al-Ayyoub, Y. Jararweh, and M. Jarrah, "Accelerating compute-intensive image segmentation algorithms using GPUs," *J. Supercomput.*, vol. 73, no. 5, pp. 1929–1951, 2017.
[49] H. Zhu, Y. Guo, M. Niu, G. Yang, and L. Jiao, "Distributed SAR image change detection based on Spark," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2015, pp. 4149–4152.

BIN LIU was born in Shaanxi, China, in 1981. He received the B.S. degree in computer science and technology from the Shaanxi University of Science and Technology, China, in 2004, the M.Sc. degree in technology with a major in parallel computing and cloud computing from Yunnan University, China, in 2010, and the Ph.D. degree in electronic and information engineering from Xi'an Jiaotong University, China, in 2014.

Since 2018, he has been an Associate Professor with the College of Information Engineering, Northwest A&F University, China, where he is currently a Postdoctoral Fellow with the College of Mechanical and Electronic Engineering. His research interests include cloud computing, parallel computing, and computer vision. He currently serves as a Reviewer for the IEEE Access, the IEEE Transactions on Computers, and the *Journal of Supercomputing*.



SONGRUI HE was born in Henan, China, in 1997. He is currently pursuing the B.S. degree in computer science and technology with Northwest A&F University, China. His research interests include parallel computing, cloud computing, and big data.

DONGJIAN HE received the B.E., M.E., and D.E., degrees in agricultural engineering from Northwest A&F University, in 1982, 1985, and 1998, respectively. He was a Lecturer with the College of Mechanical and Electronic Engineering, Northwest A&F University, from 1987 to 1992, where he was an Associate Professor, from 1992 to 1999. He is currently a Professor with the College of Mechanical and Electronic Engineering, Northwest A&F University. His research interests include computer graphics, image analysis, and machine vision. He is a member of the China Computer Federation, the Chairman of the Shaanxi Society of Image and Graphics, a Vice Chairman of the Electrical Information and Automation Committee of CSAE, and a member of the council of the Chinese Society for Agricultural Machinery.

YIN ZHANG (SM'16) is currently an Associate Professor with the School of Information and Safety Engineering, Zhongnan University of Economics and Law (ZUEL), China. He is a Wenlan Distinguished Scholar with ZUEL and a Chutian Distinguished Scholar, China. He has published more than 80 prestigious conference and journal papers, including eight ESI highly cited papers. His research interests include intelligent service computing, big data, and social networks. He is a Vice Chair of the IEEE Computer Society Big Data STC. He serves as an Editor or an Associate Editor for the IEEE Access, the IEEE Sensors Journal, and the *Journal of Information Processing Systems*. He is a Guest Editor of *Future Generation Computer Systems*, the IEEE IoT Journal, *Mobile Networks and Applications*, *Sensors*, *Multimedia Tools and Applications*, *Wireless Communications and Mobile Computing*, *Electronic Markets*, the *Journal of Medical Systems*, and *New Review of Hypermedia and Multimedia*. He also served as the Track Chair for the IEEE CSCN 2017 and a TPC Co-Chair for CloudComp 2015 and TRIDENTCOM 2017.

MOHSEN GUIZANI received the bachelor's (Hons.) and master's degrees in electrical engineering and the master's and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He was an Associate Vice President of Graduate Studies with Qatar University and the Chair of the Computer Science Department, Western Michigan University, and the Computer Science Department, University of West Florida. He held academic positions with the University of Missouri-Kansas City, University of Colorado Boulder, Syracuse University, and Kuwait University. He is currently a Professor and the Electrical and Computer Engineering Department Chair with the University of Idaho. He has authored nine books and more than 400 publications in refereed journals and conferences. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, and security and smart grid. He was selected as the Best Teaching Assistant with Syracuse University for two consecutive years. He received the Best Research Award from three institutions. He currently serves on the Editorial Board for several international technical journals. He is the Founder and the Editor-in-Chief of the *Wireless Communications and Mobile Computing Journal* (Wiley). He was a Guest Editor of a number of special issues in IEEE journals and magazines. He also served as a member, the Chair, and the General Chair for a number of international conferences. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker, from 2003 to 2005.