TYPE Methods
PUBLISHED 03 March 2023
DOI 10.3389/fdata.2023.1134946

OPEN ACCESS

EDITED BY
Minh Son Dao,
National Institute of Information and Communications Technology, Japan

REVIEWED BY
Muhamad Hilmil Muchtar Aditya Pradana,
National Institute of Information and Communications Technology, Japan
Vo Hoang Trong,
Chonnam National University, Republic of Korea
Viet Ngo Quoc,
Ho Chi Minh City Pedagogical University, Vietnam

*CORRESPONDENCE
Pham The Bao
✉ ptbao@sgu.edu.vn

SPECIALTY SECTION
This article was submitted to Data Mining and Management, a section of the journal Frontiers in Big Data

RECEIVED 31 December 2022
ACCEPTED 13 February 2023
PUBLISHED 03 March 2023

CITATION
Tran NY, Hieu HT and Bao PT (2023) A proposed scenario to improve the Ncut algorithm in segmentation. *Front. Big Data* 6:1134946.
doi: 10.3389/fdata.2023.1134946

A proposed scenario to improve the Ncut algorithm in segmentation

Nhu Y. Tran[1,2], Huynh Trung Hieu[1] and Pham The Bao[3]*

[1]Faculty of Information Technology, Industrial University of Ho Chi Minh City, Ho Chi Minh City, Vietnam, [2]Information Technology Faculty, Ho Chi Minh City University of Food Industry, Ho Chi Minh City, Vietnam, [3]Information Science Faculty, Sai Gon University, Ho Chi Minh City, Vietnam

In image segmentation, there are many methods to accomplish the result of segmenting an image into k clusters. However, the number of clusters k is always defined before running the process. It is defined by some observation or knowledge based on the application. In this paper, we propose a new scenario in order to define the value k clusters automatically using histogram information. This scenario is applied to Ncut algorithm and speeds up the running time by using CUDA language to parallel computing in GPU. The Ncut is improved in four steps: determination of number of clusters in segmentation, computing the similarity matrix W, computing the similarity matrix's eigenvalues, and grouping on the Fuzzy C-Means (FCM) clustering algorithm. Some experimental results are shown to prove that our scenario is 20 times faster than the Ncut algorithm while keeping the same accuracy.

KEYWORDS

GPU, CPU, parallel computing, Ncut, FCM

1. Introduction

Image segmentation is a key step for grouping objects which have the same characteristics or properties such as color, intensity, or texture. In an image, each object is called the domain—Region and its outlines are called the boundary. Feature vectors of regions are created based on their properties and are used to distinguish them. Image segmentation describes details of the various components in image to classify and recognize objects easier (Nock and Nielsen, 2004; Starck et al., 2005; Yu et al., 2009; Belahcene et al., 2014; Dhanachandra et al., 2015; Minaee and Wang, 2019). For instance, based on image segmentation, face detection contributes to better face recognition and user identification.

In the segmentation process of image with high-resolution data, the parallel computation on the image data is divided into two approaches. Approach to parallel programming model with hybrid model on CPU - GPU (Agulleiro et al., 2012) is a powerful co-processor system because the CPU and GPU have the combined properties of using both types of additional processors allowing for the execution of many large applications for optimal performance. Specifically, OpenMP, CUDA, and MPI libraries are used on CPUs and GPUs (Sirotković et al., 2012; Baker and Balhaf, 2017; Fakhi et al., 2017; Dalvand et al., 2020; Wang N. et al., 2020). Clustering in a multi-core architecture starts with dividing the image data into regions in a grid pattern, and then parallelizes the segmentation over the regions. Parallel file system approach with Hadoop, MarReduce, and Spark (Augustine and Raj, 2016; Li et al., 2016; Cao et al., 2018; Liu et al., 2019; Wang X. et al., 2020) is built for high resolution image segmentation. All of the above approaches allow increasing the performance of the algorithm.

Among the several segmentation algorithms, Ncut algorithm (Shi and Malik, 2000) is one of the efficient algorithms for image segmentation, which is based on graph theory. It detects the boundary between two regions by partitioning and grouping based on not only local features of image but also global features of image. In this algorithm, a distinct parameter for dividing the input image into different regions was calculated. However, in graph theory, dividing graph problem is an NP-complete problem meaning that it cannot be solved in polynomial time. Besides, the complexity of the Ncut algorithm is affected by the size of the input image. On that basis, for the purpose of improving the processing time of image segments, several parallelization methods have been applied. Shiloach and Vishkin (1982) developed the first parallel algorithm based on the breadth-first search algorithm. Anderson and Setubal (1992) use parallel in the numbering algorithm on workstations for better computation. Wassenberg et al. (2009) used the Minimum Spanning Tree with optimal function calculation and parallel execution on machines sharing memory. XianLou and ShuangYuan implemented Ncut algorithm in parallel on GPUs. Therefore, the performance of these methods depends on the size of each small area. The right number of divisions and the proper size is a problem (XianLou and ShuangYuan, 2013). However, these methods mostly implement parallel algorithms on each small partition of the image. That means the image is initially divided into many small regions and then applying the segmentation algorithm on small areas in parallel.

The execution time of Ncut algorithm is $O(MxN)$ in which N is the number of pixels that is equivalent to the number of nodes of the graph created by image. Besides, M is the number of steps that Lanczos algorithm (Cullum and Willoughby, 2002) takes to find the eigenvalues in the process. Since every node of input image only relates with some neighbors, W matrix can be stored as a sparse matrix which is efficient usage memory. Moreover, because computing the similarity matrix W and its eigenvalues take too much time, we propose a parallel computing method using CUDA on GPU for solving Ncut problem.

Our paper is organized into three main sections, the first section is the introduction of our approach, which is discussed in this paper. The second section describes our proposed method in detail. This section consists of three subsections: determination of number of clusters in segmentation, computing the similarity matrix W, computing the similarity matrix's eigenvalues, and grouping on the FCM. The last section is about some experimental results in comparing the speed time between our approach and some conventional approaches and comparison of accuracy in segmentation.

2. Proposed method

Ncut method proposed by the group author Shi and Malik is as follow:

+ Given an image, set up a weighted graph $G = (V, E)$, where the vertex set V of graph are the point in the feature space, every edge in the edge set E is formed between every pair of nodes, and set the weight on the edge connecting two nodes to be a measure of the similarity between the two nodes.
+ Solve $(D - W)x = \lambda Dx$ for eigenvector with the smallest eigenvalues. Where, D is an $N \times N$ diagonal matrix, W is an $N \times N$ symmetrical matrix, x is an eigenvector, and λ is an eigenvalue.
+ Use the eigenvector with the second smallest eigenvalue to bipartition the graph by finding the splitting point such that *Ncut* is minimized.
+ Decide if the current partition should be subdivided and recursively repartition the segment parts if necessary.
+ Recursively repartition the segment part if necessary.

The Ncut algorithm should be improved in the image segmentation problem for computational performance. Firstly, automatic k cluster prediction method is needed to choose the number of k partitions in image segmentation applications, we propose to predict the number k clusters based on the characteristic histogram of the image. Secondly, in one step of the algorithm, the K-means grouping method is used to group on the eigenvector set found. Since the eigenvector set is a real data set, there will be errors during the clustering calculation due to the computer's structure and numerical representation. We propose to use FCM algorithm with the expectation that it can fuzzify the data so that errors can be accepted for better data clustering prediction. Furthermore, the process of finding the similarity matrix and eigenvalues of the sequential execution problem takes up a considerable amount of execution time in the whole algorithm. We propose to apply parallel computation on GPU for this calculation step with the expectation of better computing performance on large image data.

2.1. Determination of number of clusters in segmentation

In most of segmentation problems, the issue of deciding the number of objects in order for segmentation is crucial and indispensable. Generally, this number of groups will

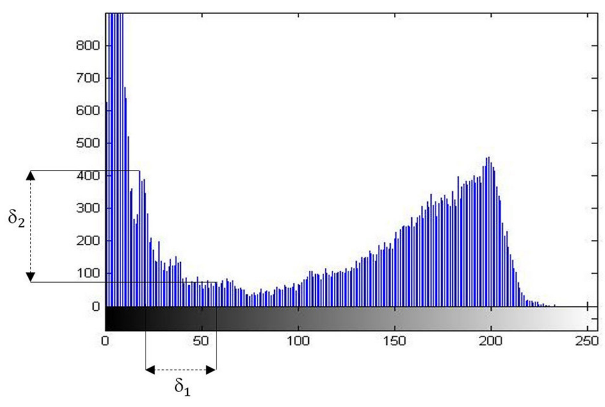

FIGURE 1
The histogram of image depicts the distance and differences.

be intuitively inputted based on user estimation. The estimation comes from viewing an arbitrary image and giving a number k.

We propose an automatic approach to deciding the number k of clusters based on histogram. The gray-level histogram provides many extreme points (minimum and maximum points). The exploration of number of maximum points is the key of deciding the number of clusters in which the following formula is satisfied (1).

Let $\delta_i \in R$ and define $f: R^2 \times R^2 \rightarrow R^2 \times R^2$ by

$$f(P_1, P_2) = \{(P_1, P_2) \mid (P_1(x) - P_2(x)) > \delta_1 \ \& \ (P_1(y) - P_2(y)) > \delta_2\} \tag{1}$$

Where $(P_1(x) - P_2(x)) > \delta_1$ is the height deviation or the deviation of the total pixels in a gray-level between two peaks. And the $(P_1(y) - P_2(y)) > \delta_2$ is the distance between two extreme points. The value of δ_1 and δ_2 is estimated by statistics from a pre-selected set of images.

The Figure 1 of gray-level histogram shows us the number of clusters based on distance and deviation.

Algorithm 1 is present to determine the number of segments in an image.

```
Input: Array is gray-level histogram, $\delta_1$, $\delta_2$
Output: K image segments
K:=0
Step 1: Find the largest of local element $P_1$
Step 2: Find the smallest of local element $P_2$ such
that $P_2(y) < P_1(y)$ and satisfy formula (1)
Step 3: K increases by 1 unit
Step 4: Repeat step 1
```

Algorithm 1. K-segment_Histogram.

2.2. Computing the similarity matrix W

The similarity matrix W or the matrix affinity is the matrix representation of the relationship between the nodes in the original image. For example, the original image is converted to graph and W matrix as shown in Figure 2.

We apply the Malik and Shi grouping algorithm to image segmentation based on brightness. We construct the graph $G = (V, E)$ by taking each pixel as a node and define the edge weight w_{ij} between node i and j as the product of a feature similarity term and spatial proximity term using formula (2).

$$w_{ij} = e^{-\frac{\|I(i)-I(j)\|_2^2}{\sigma_I}} \times \begin{cases} e^{-\frac{\|X(i)-X(j)\|_2^2}{\sigma_X}} & \text{if } \|X(i)-X(j)\|_2 < r \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Where $X(i)$ is the spatial location of node i and $I(i)$ is the intensity value of the brightness. We have the weight $w_{ij} = 0$ for any pair of nodes i and j that are more than r pixel apart.

As we see, r is often small than size of matrix image (Shi and Malik, 2000). Therefore, the numbers of zeros elements are more than other elements in the similarity matrix. In other words, matrix affinity W is a sparse matrix. To save memory usage, we stored W in form coordinate (COO) (Bell and Garland, 2008) which consist of element's indexes having nonzero elements. Specifically, the input matrix A contained into three arrays *row*, *col* and *val* corresponding with row index, column index and value of nonzero elements as shown in Figure 3.

For each pixel i, there is corresponding connected pixels j. In other words, there will be w_{ij} for two connected pixels i and j according to the given connection distance r. Thus, with the input image I(row×col), it will take a long time to consider the vertices sequentially (Shi and Malik, 2000). Therefore, we propose Algorithm 2 to parallelize each vertex i∊(row×col) to find connected vertices j and calculate the weight w_{ij} using formula (2).

A

$$\begin{bmatrix} 1 & 3 & 0 & 5 \\ 3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 \\ 5 & 0 & 4 & 1 \end{bmatrix}$$

B

$$\begin{aligned} \text{row} &= [0\ 0\ 0\ 1\ 1\ 2\ 2\ 3\ 3\ 3] \\ \text{col} &= [0\ 1\ 3\ 0\ 1\ 2\ 3\ 0\ 2\ 3] \\ \text{val} &= [1\ 3\ 5\ 3\ 1\ 1\ 4\ 5\ 4\ 1] \end{aligned}$$

FIGURE 3
The COO form of input matrix: **(A)** Input matrix. **(B)** The COO format of input matrix.

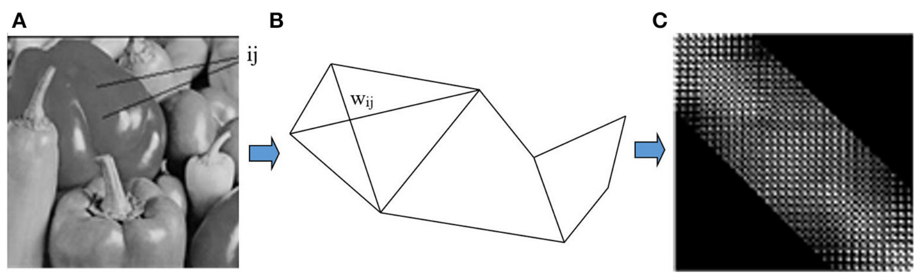

FIGURE 2
The graph and the similarity of original image: **(A)** Original image. **(B)** Grap G. **(C)** Matrix affinity W.

```
Input:Original image I; r is the radius; nrow, ncol are sizes of the input image; dMaskx, dMasky, dnMask are abscissa, ordinate and number of elements in the mask by Appendix A
Output:COO form of the W matrix (dVal, dRow, dCol)
 Step 1: Copy image I, dMaskx, dMasky, and dnMask from host (CPU memory) to device (GPU memory)
 Step 2: Determine the total number of non-zero elements of the weight matrix W corresponding to the input matrix I by Algorithm 6
 Step 3: Allocate the memory for variables dRow, dCol, dVal are $n \times sizeof(int)$, $n \times sizeof(int)$, $n \times sizeof(double)$
 Step 4: Build a grid of execution threads (gridDim = (floor(((nrow×ncol)/r)$^2$), 1, 1), blockDim = (r, r, 1))
 Step 5: Call kernel function, execute parallelly threads to calculate dVal, dCol, dRow:
  Step 5.1: Determine the index of the threads under execution
  idx = blockidx.x*blockDim.x + threadIdx.x
  Step 5.2: Calculate the pixel position under consideration: $tx = \frac{idx}{ncol}$; $ty = idx\%\ ncol$
  Step 5.3: Calculate the neighboring points at the pixel under consideration that will have positions $x = tx + dMaskx[i]$, $y = ty + dMasky[i]$ $withi \in [0,..,dnMask]$ respectively
 Step 5.4: Calculate the wij values by formula (2) and store in the variables dVal, dRow, dCol
```

Algorithm 2. Sparse matrix_Ncut_GPU.

```
Input:The symmetric matrix $A_{n\times n}$, k
Output:The k smallest eigenvalue and eigenvector
 Step 1: Random vector $v_1 \in R^n$ and $\|v_1\|_2 = 1$
 Step 2: $j = 1$, $\beta_1 = 0, v_0 = 0$
 Step 3: We compute:
 $w_j = Av_j$ by Algorithm 8
 $\alpha_j = w_j v_j$ by Algorithm 7
 $w_j = w_j - \alpha_j v_j - \beta_j v_{j-1}$ by Algorithm 9
 $\beta_{j+1} = \|w_j\|$
 $v_{j+1} = w_j/\beta_{j+1}$
  $T[j,j] = \alpha_j$
  $T[j-1,j] = \beta_{j+1}$
 $T[j,j-1] = \beta_{j+1}$
 $j = j+1$
 Step 4: If $j < k-1$ back to Step 4, else continue
 Step 5: Calculate the k smallest eigenvalue and eigenvector based on diagonal matrix $T$
```

Algorithm 3. Lanczos_Ncut_GPU.

2.3. Computing the similarity matrix's eigenvalues

The Ncut problem to find eigenvalues of weight matrix W means finding k smallest eigenvectors of Laplace matrix based on weight matrix W. Since weight matrix W is a symmetric sparse matrix of relatively large size, it takes a lot of time to compute the eigenvector in the whole image segmentation process. Especially with the larger image size, the matrix size also increases exponentially, so it takes more time to find the eigenvalues in this case. We propose a method to parallelize the eigenvalues of the W matrix on the GPU and the Lanczos method (Cullum and Willoughby, 2002) as an effective algorithm for finding the k smallest eigenvalues. We compute Lanczos by paralleling in GPU. Each j, we compute parallel the multiplication between matrix and vector, the multiplication between vector and vector according to Algorithm 3.

2.4. Applying FCM algorithm to eigenvector matrix

The FCM algorithm (Nayak et al., 2014) allows a point to belong to one or more groups depending on the degree of membership function of each point corresponding to the centers of the groups (using fuzzy logic). Therefore, FCM has more flexibility with data sets with overlapping data clusters (high similarity with images). The algorithm is mainly based on the optimization of the objective function according to the formula (3).

$$J_{fcm}(Z,U,V) = \sum_{i=1}^{C}\sum_{j=1}^{N} u_{ij}^{m}\left\|x_j - c_i\right\|^2 \tag{3}$$

In which, the matrix belonging to $U=[u_{ij}] \in M_{fcm}$ is the fuzzy partition of the data set Z, $u_{ij} \in [0,1]$ indicates the dependence of point x_i on the jth cluster, with $\sum_{j=1}^{C} u_{ij} = 1, \forall i$; $V = [c_1, c_2, \ldots, c_c]$ is the sample vector or cluster center of the C groups, calculated according to the distance standard $D_{ij} = \left\|x_j^i - c\right\|^2$; $m \in [1;\infty]$ is the exponent that determines the fuzziness of the clustering.

Instead of using k-mean in Ncut algorithm, we propose the FCM algorithm to group the image from second smallest eigenvalue by fuzzy original data to optimal cluster according to Algorithm 4. The FCM are fuzzy-graph structures for representing data in a fuzzy way. It accepted a computation noise for clustering, and it is more optimal on the real data, and eigenvector set from the Ncut problem.

3. Experimental result

We setup the algorithm in the personal computer which had the Intel core i5 CPU (2.3 GHz), 8 GB RAM, GeForce GT 1050Ti, window 10 professional 64. We coded parallel in CUDA (CUDA Toolkit v9.0 and visual studio 10.0) and then implemented in MATLAB (R2016b). Our data (http://decsai.ugr.es/cvg/dbimagenes) included 15 images 128 × 128, 37 images 256 × 256, 64 images 512 × 512, and 3 images 1,024 × 1,024.

3.1. Determination of number of clusters in segmentation

We conduct an experiment of determination of number of clusters in segmentation on four datasets. The comparison is processed on the resultant number of clusters from intuition and our method on histogram. The Table 1 is the demonstration of our experiment on 4 datasets with the threshold of height deviation and peak distance of δ_1 and δ_2 as 50 and 100 respectively.

From Table 1, it can be seen that the difference between the two ways of visual determination and that using histogram is not significant. Therefore, the prediction of cluster number is feasible in dynamic image processing applications.

```
Input:k eigenvector matrix
Output:Clustered k eigenvector matrix
 Step 1: Let the matrix U $\in R^{nxk}$ be the matrix
 with k eigenvectors $v_1, ..., v_k$ in column form
 Step 2: For i = 1, ..., n, let $y_i \in R^k$ be
the vector corresponding to the ith row of the
 matrix U
 Step 3: Group the points $(y_i)_{i=1,...,n} \in R^k$ using FCM
into clusters $C_1, ..., C_k$
 Step 4: We have the result as clusters $A_1, ...,$
 $A_k$ with $A_i = \{j/ y_j \in C_i\}$
```

Algorithm 4. FCM_Ncut.

TABLE 1 The number of clusters from an image by intuition and K-segment_Histogram algorithm on using histogram.

Data set	Number of correct images	Number of false images	Average number of different clusters
1	11	4	1.5
2	31	6	1.3
3	57	7	1.5
4	3	0	0

TABLE 2 The demonstration of speed time to calculate matrix W between Shi/Malik and sparse matrix_Ncut_GPU algorithm.

Size	Runtime(s)	
	Shi/Malik	Sparse matrix_Ncut_GPU
128 x 128	0.57	0.16
140 x 140	0.66	0.03
200 x 200	1.41	0.04
256 x 256	5.13	0.05
512 x 512	42.13	0.13
1,024 x 1,024	NA	0.43

TABLE 3 The demonstration of speed time to calculate eigenvalues between Shi/Malik method and Lanczos_Ncut_GPU method.

Size	Runtime(s)	
	Calculating eigenvalues in the Shi/Malik algorithm	Calculating eigenvalues in the Lanczos_Ncut_GPU algorithm
128 x 128	1.63	0.69
140 x 140	2.36	0.85
200 x 200	6.21	0.94
256 x 256	17.34	0.97
512 x 512	NA	1.28
1,024 x 1,024	NA	2.84

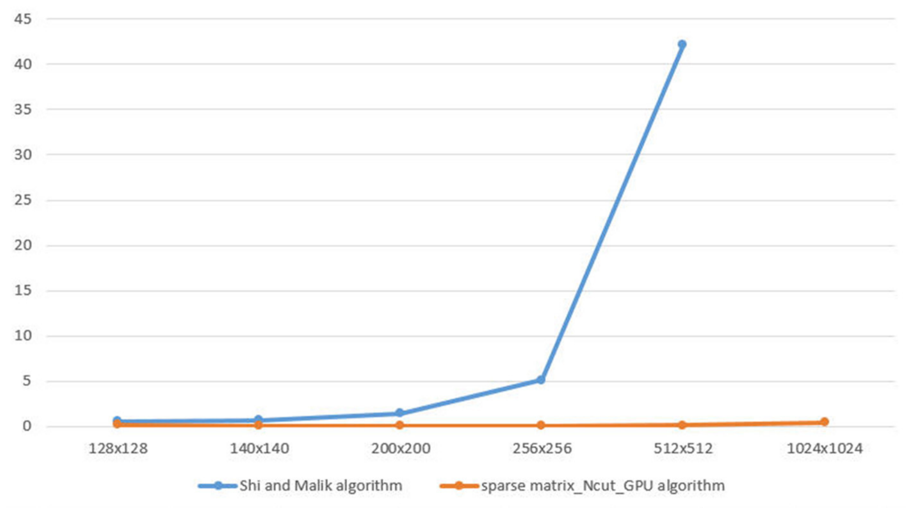

FIGURE 4
The speed time between Shi/Malik algorithm and sparse matrix_Ncut_GPU algorithm.

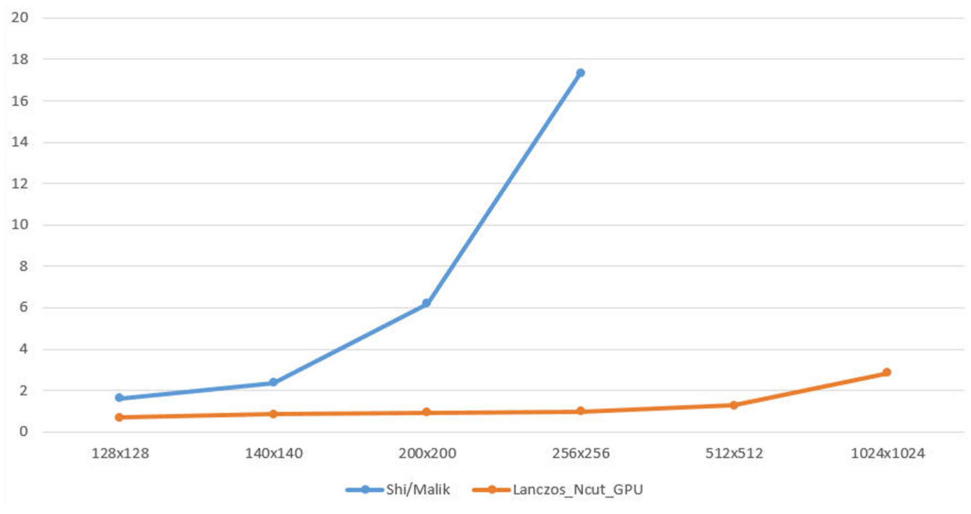

FIGURE 5
Calculating eigenvalues in the Shi/Malik method and Lanczos_Ncut_GPU method with size of image (3 segment).

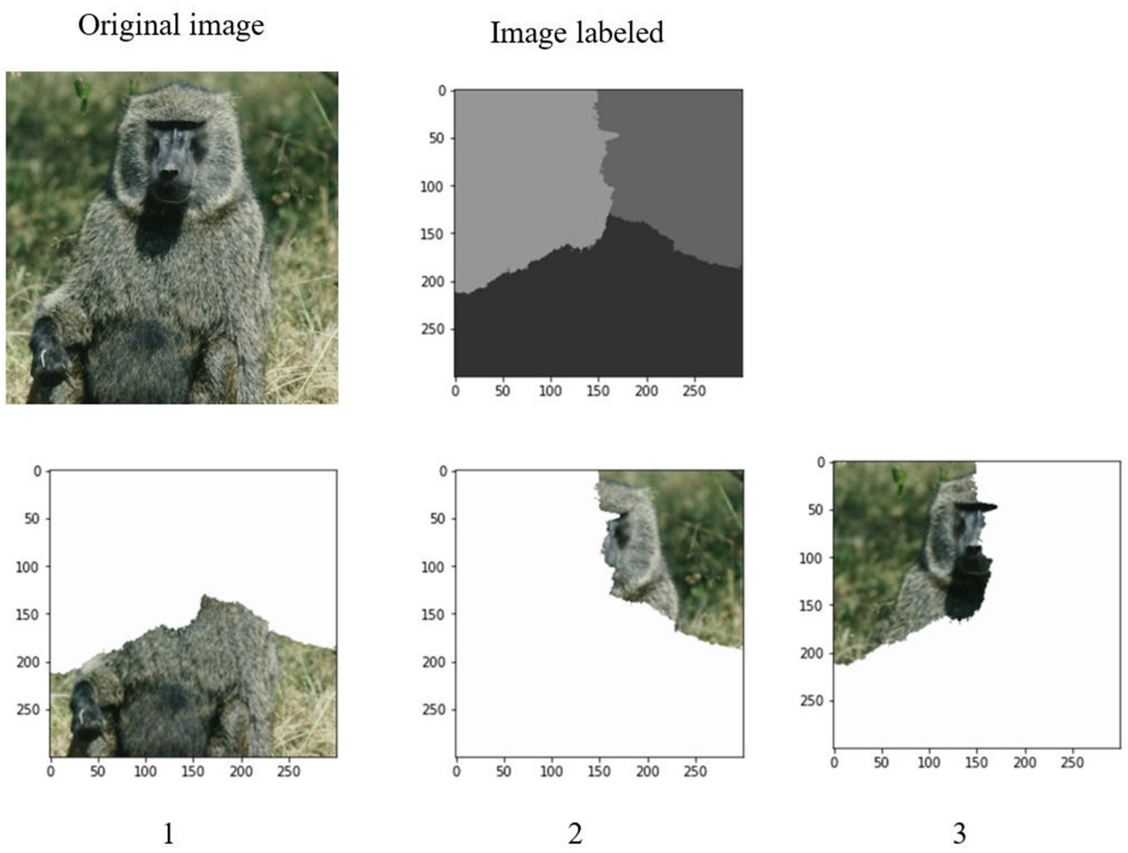

FIGURE 6
Grouping on the eigenvalue by k-mean to k classes.

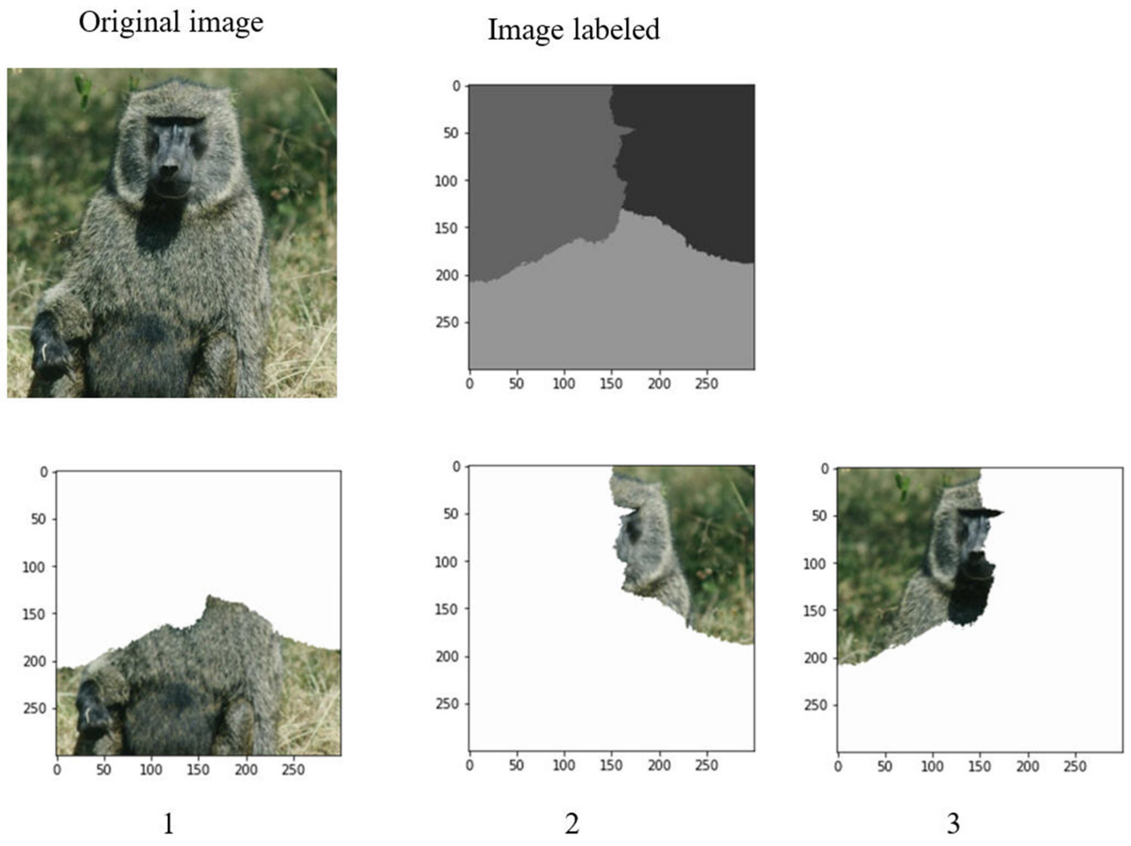

FIGURE 7
Grouping on the eigenvalue by FCM to k classes.

3.2. Computing the similarity matrix W

As Figure 4, we have the graph illustrates the speed time to calculate W matrix between Shi and Malik algorithm and sparse matrix_Ncut_GPU algorithm.

The vertical axis is the speed(s) and the horizontal axis is the size of image which is demonstrated in Table 2.

In the graph, when image is 1024 × 1024, the Shi and Malik method is being out of memory. The sparse matrix_Ncut_GPU algorithm executes for a small increase when the image size increases.

3.3. Computing the matrix's eigenvalues

In this section, the experiment is conducted on the increasing image size in Table 3 on the three clusters ($k = 3$). This corresponds to exploring three eigenvalue and eigenvector of the W matrix. The Table 3 represents us that the GPU time (time to find eigenvalue in parallel way) is quicker than Shi/Malik method.

The graph below illustrate the Shi/Malik method and Lanczos_Ncut_GPU method with size of image, the number of segment as shown in Figure 5.

The result of calculating eigenvalues by Lanczos_Ncut_GPU algorithm is 17 times faster than that by Shi/Malik algorithm when the image size increases.

3.4. Grouping on the eigenvalue by k−mean and FCM

After determination of a set of eigenvalues, the Ncut algorithm makes use of these eigenvalues to cluster them into k clusters. It corresponds to the k regions of an image. Figures 6, 7 demonstrate the clustering results by K-means and FCM. There are no differences between the two approaches to image segmentation. Figure 8 and Table 4 depict the computational time in image segmentation between Shi/Malik's algorithm and the one we improved by GPU. It is found that the execution time of the algorithm we improved by GPU is 20 times faster than that of the Shi/Malik algorithm while keeping the same accuracy.

4. Conclusion

In this paper, we analyze the Ncut algorithm to cluster an image into regions. Because in the Ncut algorithm, we have to input the number of clusters k, it is not automated process. For that

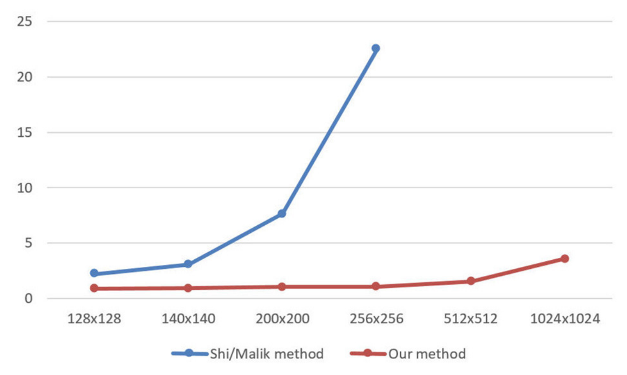

FIGURE 8
The speed time to calculate image segmentations between Shi/Malik algorithm and Our algorithm.

TABLE 4 The demonstration of speed time to calculate image segmentations between Shi/Malik method and our method by GPU.

Size	Runtime(s)	
	Shi/Malik	Sparse matrix_Ncut_GPU
128 x 128	2.22	0.88
140 x 140	3.04	0.92
200 x 200	7.67	1.04
256 x 256	22.51	1.09
512 x 512	NA	1.56
1,024 x 1,024	NA	3.58

reason, we predict k group depending on histogram will give a good result. Moreover, we improve the speed performance by parallel computing on GPU. Specifically, it is to parallelize the calculation of the W matrix and finding of the eigenvector matrix in the Ncut algorithm. Finally, we used the FCM algorithm to cluster the data in the above eigenvector matrix. Some experimental results on image data sets are conducted to prove that our approach is 20 times quicker than Shi and Malik approach in computing eigenvalue and computing similarity matrix.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found here: https://ccia.ugr.es/cvg/dbimagenes.

Author contributions

PB, HH, and NT have designed methods. NT implemented it and tested performance. All authors contributed to the article and approved the submitted version.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fdata.2023.1134946/full#supplementary-material

References

Agulleiro, J. I., Vázquez, F., Garzón, E. M., and Fernández, J. J. (2012). Hybrid computing: CPU+GPU co-processing and its application to tomographic reconstruction. *Ultramicroscopy*. 115, 109–114. doi: 10.1016/j.ultramic.2012.02.003

Anderson, R. J., and Setubal, J. (1992). "On the parallel implementation of goldbergs maximum flow aglrotithm," in *Proceedings of the 4th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)* (Brazil), 168–177.

Augustine, D. P., and Raj, P. (2016). Performance evaluation of parallel genetic algorithm for brain MRI segmentation in hadoop and spark. *Indian J. Sci. Technol.* 9, 1373. doi: 10.17485/ijst/2016/v9i48/91373

Baker, Q. B., and Balhaf, K. (2017). "Exploiting GPUs to accelerate white blood cells segmentation in microscopic blood images," in *Proceedings of the 8th International Conference on Information and Communication Systems (ICICS)* (Irbid. Jordan), 136–140.

Belahcene, M., Chouchane, A., Benatia, M. A., and Halitim, M. (2014). "3D and 2D face recognition based on image segmentation," in *Proceedings of the 2014 International Workshop on Computational Intelligence for Multimedia Understanding (IWCIM)* (Paris, France).

Bell, E., and Garland, M. (2008). Efficient sparse matrix-vector multiplication on CUDA. *NVIDIA Technical Report* NVR-2008–R-2004.

Cao, J., Chen, L., Wang, M., and Tian, Y. (2018). Implementing a parallel image edge detection algorithm based on the otsu-canny operator on the hadoop platform. *Comput. Intell. Neurosci.* 20, 8284. doi: 10.1155/2018/3598284

Cullum, J. K., and Willoughby, R. A. (2002). *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*. SIAM: Society for Industrial and Applied Mathematics, Birkhäuser, Boston, United States.

Dalvand, M., Fathi, A., and Kamran, A. (2020). Flooding region growing: a new parallel image segmentation model based on membrane computing. *J. Real-Time Image Process*. 18, 37–55. doi: 10.1007/s11554-020-00949-0

Dhanachandra, N., Manglem, K., and Chanu, Y. J. (2015). Image segmentation using k-means clustering algorithm and subtractive clustering algorithm. *Procedia Comput. Sci.* 54, 764–771. doi: 10.1016/j.procs.2015.06.090

Fakhi, H., Bouattane, O., Youssfi, M., and Hassan, O. (2017). "New optimized GPU version of the k-means algorithm for large-sized image segmentation," in *Proceedings of the 2017 Intelligent Systems and Computer Vision (ISCV)* (Fez, Morocc).

Li, X., Song, J., Zhang, F., Ouyang, X., and Khan, S. U. (2016). Mapreduce-based fast fuzzy C-means algorithm for large-scale underwater image segmentation. *Fut. Gen. Comp. Sys*. 65, 90–101. doi: 10.1016/j.future.2016.03.004

Liu, B., He, S., He, D., Zhang, Y., and Guizani, M. (2019). A spark-based parallel fuzzy C-means segmentation algorithm for agricultural image big data. *IEEE access*. 7, 42169–42180. doi: 10.1109/ACCESS.2019.2907573

Minaee, S., and Wang, Y. (2019). An admm approach to masked signal decomposition using subspace representation. *IEEE Transact. Image Process*. 28, 3192–3204. doi: 10.1109/TIP.2019.2894966

Nayak, J., Naik, B., and Behera, H. S. (2014). "Fuzzy C-Means (FCM) Clustering algorithm: a decade review from 2000 to 2014," in *Proceedings of the Computational Intelligence in Data Mining. Smart Innovation, Systems and Technologie* (Springer, New Delhi), 2 133–149. doi: 10.1007/978-81-322-2208-8_14

Nock, R., and Nielsen, F. (2004). Statistical region merging. *IEEE Trans. Pattern Anal. Mach. Intell*. 26, 1452–1458. doi: 10.1109/TPAMI.2004.110

Shi, J., and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell*. 22, 888–905. doi: 10.1109/34.868688

Shiloach, Y., and Vishkin, U. (1982). An o(n^2log n) parallel max-flow algorithm. *J. Algor*. 3, 128–146. doi: 10.1016/0196-6774(82)90013-X

Sirotković, J., Dujmić, H., and Papić, V. (2012). "K-means image segmentation on massively parallel GPU architecture," in *Proceedings of the 35th International Convention MIPRO* (Opatija. Croatia), 489–494.

Starck, J.-L., Elad, M., and Donoho, D. L. (2005). Image decomposition via the combination of sparse representations and a variational approach. *IEEE Transactions on Image Processing*. 14, 1570–1582. doi: 10.1109/TIP.2005.852206

Wang, N., Chen, F., Yu, B., and Qin, Y. (2020). Segmentation of large-scale remotely sensed images on a Spark platform: a strategy for handling massive image tiles with the MapReduce model. *ISPRS J. Photogram. emote Sens*. 162, 137–147. doi: 10.1016/j.isprsjprs.2020.02.012

Wang, X., Pan, J. S., and Chu, S. C. (2020). A parallel multi-verse optimizer for application in multilevel image segmentation. *IEEE Access*. 8, 32018–32030. doi: 10.1109/ACCESS.2020.2973411

Wassenberg, J., Middelmann, W., and Sanders, P. (2009). "An Efficient parallel Algorithm for Graph-based Image segmentaion," in *Proceedings of the 2009 Computer Analysis of Images and Patterns* (Springer, Berlin. Heidelberg), 1003–1010.

XianLou, H., and ShuangYuan, Y. (2013). "Image segmentation based on Normalized Cut and CUDA parallel implementation," in *Proceedings of the 5th IET International Conference on Wireless, Mobile and Multimedia Networks (ICWMMN 2013)* (Beijing China) 209–214.

Yu, T., Lim, S. N., Patwardhan, K., and Krahnstoever, N. (2009). "Monitoring, recognizing and discovering social networks," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition* (Miami, FL), 1462–1469.