A Parallel Approach for Image Segmentation using Fuzzy C-Means

Gonzalo Alvis
Facultad de Ingeniería
Universidad del Pacífico
Lima, Perú
gr.alvisb@alum.up.edu.pe

César Cabezas
Facultad de Ingeniería
Universidad del Pacífico
Lima, Perú
cm.cabezasg@alum.up.edu.pe

Franz Figueroa
Facultad de Ingeniería
Universidad del Pacífico
Lima, Perú
f.figueroag@alum.up.edu.pe

Miguel Muñoz
Facultad de Ingeniería
Universidad del Pacífico
Lima, Perú
ma.muñozc@alum.up.edu.pe

***Abstract*—This work addresses the optimization of the Fuzzy C-Means (FCM) algorithm for the segmentation of blood cell images by parallelizing it with OpenMP in C++. FCM, known for its effectiveness in handling data with classification ambiguities, faces performance challenges when scaling to large volumes of data, such as high-resolution images. To mitigate the extensive processing times associated with the algorithm in its traditional form, we propose parallelizing the algorithm. A parallel implementation of FCM was developed, allowing its simultaneous execution on multiple processor cores, aiming to improve processing efficiency without compromising segmentation accuracy. The methodology includes the parts of the algorithm adapted for parallelization, the preparation and segmentation of the selected set of cell images, and the performance evaluation through comparisons with the sequential version of the algorithm. The results demonstrated significant improvements in processing times, with notable speedups as the number of threads used increased. However, a decrease in efficiency was observed with an increase in threads, suggesting a balance between the number of threads and the size of the dataset to optimize performance.**

***Index Terms*—Fuzzy C-Means, OpenMP, Parallelization**

I. Introduction

The analysis of large volumes of data has gained significant importance in fields such as biomedicine, economics, and engineering [1]. One of the most prominent techniques for data analysis is clustering, a process that groups data according to a similarity measure, such as Euclidean distance [2]. This technique has applications in areas related to pattern recognition, such as image segmentation, which divides an image into uniform regions based on similarity measures between pixels [3]. However, the scalability of clustering algorithms poses an increasing challenge as data volumes or the number of clusters grow [4].

Fuzzy C-Means (FCM) is one of the most widely used clustering methods, particularly effective in handling data with ambiguity in class definitions [5]. However, FCM is computationally expensive, limiting its application to massive datasets [6]. Nevertheless, some processes of the algorithm can be executed simultaneously, offering an opportunity to improve its efficiency without compromising accuracy.

Parallelization at the hardware level, through multi-core processing based on high-performance computing (HPC) and tools such as the OpenMP (OMP) library, allows for optimal use of modern hardware resources and a significant reduction in execution times [7]. This improvement is crucial in real-time applications and the analysis of large data volumes. In the field of biomedicine, one of the main challenges is the implementation of machine learning models for early diagnosis through the analysis of laboratory samples [8], with the goal of recognizing viruses or harmful cells to diagnose diseases such as tuberculosis or various types of cancer.

The time required for laboratory test diagnosis under a microscope varies according to the patient's background. For instance, a sputum test to rule out tuberculosis can take from minutes to hours, while a cancer recognition test may require several hours, demanding a more detailed review of the samples [9].

The general objective of this project is to develop and optimize a parallel version of the Fuzzy C-Means algorithm for image segmentation using OpenMP in C++. This implementation aims to significantly improve performance in terms of execution time and computational efficiency, ensuring segmentation accuracy. Additionally, an analysis will be conducted using various performance metrics such as speedup and efficiency in relation to changes in different parameters.

The relevance of this work lies not only in the crucial role these techniques have acquired in the field of medicine but also in the efficiency results demonstrated in the development of diagnostics [10], applying fuzzy clustering techniques in the field of X-rays. Reducing computational costs while maintaining the effectiveness of training new models preserves the accessibility of this tool, reducing waiting times and technical requirements for its application. This not only encourages its use but also promotes experimentation for advancement in this field.

Our motivation is to facilitate the development and application of pattern recognition tools in images through segmentation, analyzing the process architecture with HPC tools like OMP. This approach aims to improve the efficiency and accessibility of these techniques, promoting their adoption and advancement in various application fields.

Regarding specific objectives, the following are considered:

1) Perform a detailed analysis of the Fuzzy C-Means algorithm, identifying the parts of the algorithm that are most suitable for parallelization, with a specific focus on its application for segmenting digital images.

2) Develop a parallel version of the algorithm by applying OpenMP clauses to optimize image processing. Additionally, ensure that the segmentation accuracy is maintained using metrics derived from the masks that each image in the database has.
3) Compare the performance of the parallelized version of the algorithm with its sequential version, using different parameter configurations such as the number of threads, number of iterations, image dimensions, and number of clusters to evaluate improvements in processing time and operational efficiency.

According to the reviewed literature, the parallelization and performance of the algorithm are important factors in the field of image segmentation. Vela-Rincón et al. [11] propose a variation of FCM for image segmentation based on hesitant fuzzy set theory, which allows for multiple and variable membership assignment, providing a more flexible and accurate tool for image analysis compared to traditional fuzzy set-based methods. The authors implemented a hardware-level parallelization technique using OpenMP, which significantly accelerates the algorithm's processing time. Experimental results highlight the superiority of this algorithm in preserving details at the edges of segmented regions and in the homogeneity of these regions.

In the study by Vikraman and Afthab [12], an optimized Fuzzy C-Means approach using the Rain Optimization Algorithm (ROA) for medical image segmentation is presented. This study addresses the limitations of FCM, such as its sensitivity to initial values and exposure to noise, by proposing a noise reduction technique and a hybrid filter to improve segmentation quality. Implemented in MATLAB, the method detects and extracts tumors from brain MRI images from the BraTS dataset. Compared to other cluster-based segmentation methods, the proposed system shows significantly superior performance in lesion extraction, evidenced by evaluation metrics such as the partition coefficient (PC) and partition entropy (PE).

Finally, Pérez-Ortega et al. [13] propose the POFCM (Parallel Optimization Fuzzy C-Means) algorithm, a parallelized version of the HOFCM (Hybrid OK-Means Fuzzy C-Means) algorithm using the OpenMP technique. Its implementation is designed to handle large datasets, such as those in the realm of Big Data. The algorithm shows efficient results in terms of acceleration and parallel efficiency, highlighting its scalability capability. Experiments conducted with real and synthetic datasets show a significant reduction in solution time, achieving speedup values of up to 3.96 and parallel efficiency of 0.99. This approach is particularly useful for the segmentation of large volumes of data and is not limited to a specific domain.

The aforementioned works share a common approach with our research in terms of improving image segmentation techniques by incorporating parallel processing into the Fuzzy C-Means (FCM) algorithm. Similar to Vela-Rincón et al. [11], Vikraman and Afthab [12], and Pérez-Ortega et al. [13], our study also seeks to leverage the capabilities of parallelization to efficiently process images. Although our approach focuses on using the traditional FCM algorithm, we aim to follow a rigorous method of various experiments to evaluate not only computational performance metrics but also the quality of the obtained segmentation. With the correct segmentation masks available in our dataset, we intend to verify the accuracy of our segmented images by comparing them with the reference masks to determine the effectiveness of our approach in accurate image segmentation.

II. Theoretical Framework

Clustering is an unsupervised classification technique widely used in data mining. It divides data elements into groups, such that the elements of a group are very similar to each other, while differing from the elements of other groups [14]. Clustering can be classified into two main types: hard clustering and fuzzy clustering. In hard clustering, each element is exclusively assigned to a single group. On the other hand, in fuzzy clustering, elements can belong to multiple groups simultaneously, with a degree of membership expressed as a fraction in the interval [0,1]. This feature allows greater flexibility in data grouping, as a single element can share characteristics with multiple groups [15].

A. Fuzzy C-Means

In the specific case of FCM, according to Xu and Wunsch [16], the Fuzzy C-Means algorithm seeks to find a partition, represented as c fuzzy clusters, for a set of data objects by minimizing the cost function:

$$J(\mathbf{U},\mathbf{M}) = \sum_{i=1}^{c}\sum_{j=1}^{N}(u_{ij})^m D_{ij}^2, \tag{1}$$

where:

- $\mathbf{U} = [u_{ij}]_{c\times N}$ is the fuzzy partition matrix and $u_{ij} \in [0,1]$ is the membership coefficient of the j^{th} object in the i^{th} cluster.
- $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_c]$ is the matrix of prototypes or centroids of the clusters.
- $m \in [1,\infty)$ is the fuzzification parameter and a higher value of m favors fuzzier clusters.
- $D_{ij} = D(\mathbf{x}_j, \mathbf{m}_i)$ is the distance between $\mathbf{x}_j$ and $\mathbf{m}_i$.

The steps of the traditional FCM algorithm are summarized below:

1) Select appropriate values for m, c, and a small positive number ϵ. Initialize the centroid matrix M randomly.
2) Update the membership matrix U by:

$$u_{ij}^{(t+1)} = \frac{1}{\left(\sum_{l=1}^{c}\left(\frac{D_{ij}}{D_{lj}}\right)^{\frac{2}{m-1}}\right)} \tag{2}$$

3) Update the prototype matrix M by:

$$m_i^{(t+1)} = \frac{\sum_{j=1}^{N}\left(u_{ij}^{(t+1)}\right)^m x_j}{\sum_{j=1}^{N}\left(u_{ij}^{(t+1)}\right)^m}, \text{ for } i = 1,\ldots,c; \tag{3}$$

4) Repeat steps 2 and 3 until $\|M^{(t+1)} - M^{(t)}\| < \epsilon$.

B. Parallel Computing

In the field of computing, the design of efficient algorithms is essential for solving complex problems. Algorithms can be classified into two main categories: sequential and parallel [17].

Sequential algorithms execute instructions linearly, processing one operation at a time. This traditional approach is effective for many tasks but can become inefficient when handling large volumes of data or intensive computations. On the other hand, parallel algorithms leverage the capability of modern systems to perform multiple operations simultaneously. These algorithms divide the problem into smaller subtasks or groups of data that can be executed concurrently by different processing units [18].

To evaluate the performance of parallel algorithms, two main indicators are used: speedup and efficiency. Speedup is defined as the ratio between the execution time of a serial program and that of the parallel program performing the same work.

$$\text{Speedup} = \frac{\text{Serial Execution Time}}{\text{Parallel Execution Time}} \tag{4}$$

Efficiency measures, on average, what portion of the speedup increase is due to each of the processing units used. In other words, efficiency can be understood as the average contribution of each processing unit to the speedup increase.

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{Number of Processing Units}} \tag{5}$$

One of the important goals in optimizing parallel algorithms is to achieve speedup as close as possible to the number of cores.

C. Otsu

The Otsu method is a popular technique for automatic thresholding based on histograms. It proposes an optimization criterion to find the threshold that minimizes the intra-class variance and maximizes the inter-class variance [19]. This method is unsupervised, making it suitable for a wide variety of applications.

The Otsu algorithm can be summarized in the following steps:

1) Calculate the histogram of the image and normalize it to obtain the probability of each intensity level.
2) Calculate the global mean of the image.
3) Iterate over all possible thresholds t to find the one that minimizes the weighted sum of intra-class variances:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t), \tag{6}$$

where $q_1(t)$ and $q_2(t)$ are the cumulative probabilities of the classes C_1 and C_2 separated by the threshold t, and $\sigma_1^2(t)$ and $\sigma_2^2(t)$ are the variances of those classes.
4) Select the threshold t that minimizes $\sigma_w^2(t)$.

D. Intersection Over Union

The Intersection over Union (IoU) metric is used to evaluate the accuracy of segmentation and object detection algorithms [20]. IoU is defined as the overlap area between the prediction and the ground truth, divided by the union area of both:

$$\text{IoU} = \frac{\text{Overlap Area}}{\text{Union Area}}.$$

The IoU metric ranges from 0 to 1, where 1 indicates a perfect match between the prediction and the ground truth. Higher IoU values indicate greater accuracy of the segmentation or detection algorithm.

III. Methodology

A. Stage I: Adapting the Fuzzy C-Means Algorithm in C++

The first phase focuses on creating a robust sequential version of the algorithm in C++. This initial implementation serves as a benchmark for evaluating the performance and accuracy of subsequent versions.

Once the sequential base is established, the next stage involves adapting the algorithm for parallel execution. This process uses OpenMP, a tool designed for parallel programming in shared memory systems. Transforming the sequential code to parallel requires an analysis to identify sections that can benefit from concurrent execution, such as the membership calculation loops for each cluster and centroid updates.

Parallelism is implemented in the membership calculation and centroid updates, as seen in equations (2) and (3). For the membership calculations, *for* and *collapse* constructs are used, and for the centroid updates, the *for* construct and a critical region for the summations are employed.

After parallelization, a verification and validation phase is carried out. This step is essential to ensure that the parallel version of the algorithm produces results consistent with the original sequential implementation. Tests are conducted with various datasets to compare not only the accuracy of the results but also the stability and convergence of the algorithm under different parallel execution conditions.

B. Stage II: Data Acquisition and Preparation

In the second stage, a database of microscope images of blood cells [21] is used. This consists of 2656 images of 1600x1200 pixels each. The collection is structured so that half of the images correspond to direct samples of blood cells, while the other half consists of masks corresponding to each of these samples. These masks are essential for the precise segmentation and analysis of individual cells within each sample.

These images are used in hematological examinations such as complete blood counts for diagnosing various conditions like Anemia, Leukemia, or coagulation disorders.

C. Stage III: Image Segmentation

The third stage of the study focuses on the application and evaluation of the optimized fuzzy C-means algorithm. First, the algorithm is applied to grayscale images of blood cells. This initial segmentation helps to identify and separate different cellular components based on pixel intensity. Subsequently, the segmented images undergo a binarization process using tl Otsu thresholding method.

To evaluate the segmentation accuracy, the obtained resul are compared with the reference masks provided in the origin database. For this purpose, the Intersection over Union (Iol metric is used, providing a quantitative measure of the ove lap between the segmented masks and the reference mask thereby determining the effectiveness of the segmentatio process.

D. Stage IV: Efficiency and Speedup Measurement

The fourth stage focused on measuring the speedup a efficiency of the parallel algorithm. A series of systemat experiments were designed under various execution conc tions, where two important factors were evaluated: the numb of images processed and the number of threads used the parallelization. These experiments allow analyzing tl scalability of the algorithm and determining the optimal thre configuration to maximize performance.

IV. Results

Experiments were conducted with different quantities of images and threads, maintaining configurations with a maximum of 150 iterations and the use of 2 clusters.

A. Segmentation Masks

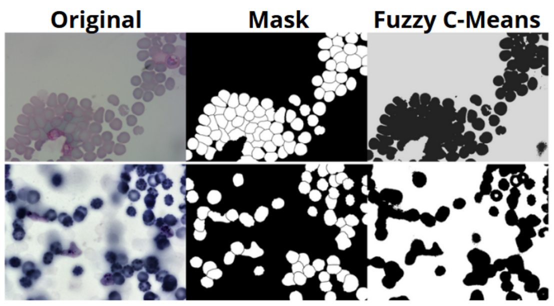

Fig. 1. Image Segmentation: Original vs. Mask vs. Result

Figure 1 shows a visual comparison of different stages in the image segmentation process using the Fuzzy C-Means method. In both rows, three columns are presented: the original image, the binary mask, and the result of the Fuzzy C-Means algorithm.

The first column contains the original images of biological samples, highlighting cellular structures. The second column displays the binary masks, where the cells are highlighted in white on a black background, facilitating the identification of areas of interest. The third column shows the results of the Fuzzy C-Means algorithm, where the cells are grouped into different clusters represented by various shades of gray, allowing for more detailed and flexible segmentation.

This comparison demonstrates the performance of the Fuzzy C-Means method in segmentation, showing that it successfully identifies the majority of cells in a visually evident manner.

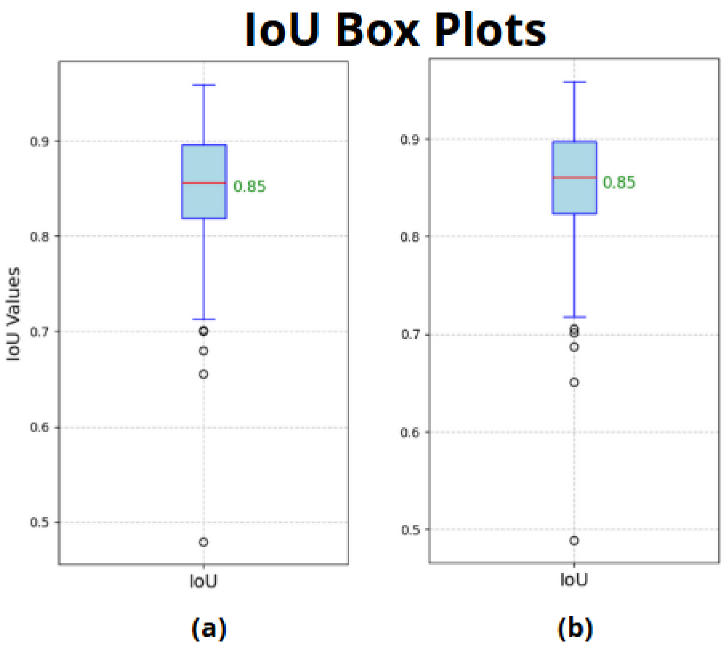

Fig. 2. Box plot comparing the Intersection over Union (IoU) index between two image segmentation methods: (a) Results from our parallel implementation of Fuzzy C-Means and (b) Results obtained using the K-means implementation from the Python library.

To quantitatively evaluate the results, biological images were segmented using the Fuzzy C-Means method, followed by thresholding with the Otsu method. According to Figure 2(a), the average Intersection over Union (IoU) value is 0.8512, calculated over 159 test images, highlighting the precision of the method in segmenting biological images. Although Figure 2(a) shows four observations with values below 0.7, the average IoU value indicates that Fuzzy C-Means achieves high concordance with the image masks.

Additionally, we compared the results obtained with the K-means implementation from the Sklearn library. As shown in Figure 2(b), the results are very similar. The exact IoU value for K-means is 0.85357, which represents an increase of 0.277% compared to the Fuzzy C-Means method.

Regarding execution times, 10 tests were conducted on a set of 10 images, comparing a Fuzzy C-Means implementation in Python with our C++ implementation. The results are presented in Figure 3. These results show that the C++ implementation using 6 threads is faster than the traditional Python implementation. Although the C++ version shows greater dispersion in execution times, its overall performance is superior.

Figure 3 illustrates the distribution of execution times for both implementations. It is notable that, despite the observed variability in the C++ version, it consistently outperforms the Python implementation in terms of speed. This analysis is based on a total of 100 executions for each language (10 tests, each with 10 images), providing a robust sample for comparison.

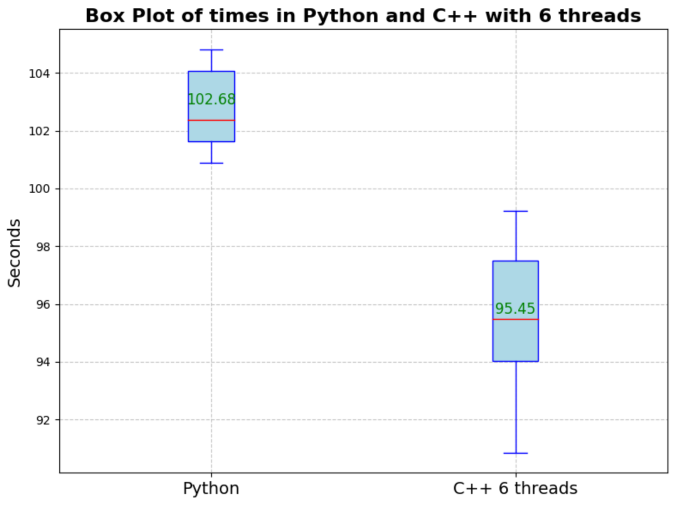

Fig. 3. Comparison of execution times between Fuzzy C-Means implementations in Python and C++ (6 threads).

This demonstrates its effectiveness in identifying and delineating cellular structures. These results reflect the method's ability to maintain consistent performance and higher speed across a diverse set of images, consolidating its applicability in biological image segmentation tasks.

B. *Parallelization Indicators*

A detailed analysis of the performance of parallelization in image processing was conducted, varying the number of threads and the size of the image sets processed simultaneously. The sizes of the evaluated sets were 10, 25, 50, 90, and 159 images. The obtained results are visualized in two graphs that provide the speedup and efficiency values. Table I shows the processing times in seconds as a function of the number of images and threads used.

TABLE I
EXECUTION TIMES IN SECONDS AS A FUNCTION OF THE NUMBER OF THREADS AND IMAGES PROCESSED

Threads/Images	10	25	50	90	159
1	163.01	398.03	821.2	1493.92	2491.231
2	106.14	251.09	561.1	970.23	1589.04
4	75.13	179.59	424.61	721.56	1179.06
6	65.89	172.98	371.79	635.32	1069.02
8	58.89	158.02	323.34	596.89	984.06

In Figure 4 and Table II, it can be observed that as the number of threads increases from 2 to 8, there is an increase in speedup for all tests conducted with different image set sizes (10, 25, 50, 90, 159). This increase is more notable when using 8 threads, where the speedup reaches a maximum of 2.255 for 10 images and remains above 1.98 for 159 images. Additionally, the speedup is higher for the 10-image set compared to larger sets like 159 images. However, the marginal growth in speedup with the increase in threads begins to decrease, which is further analyzed in the efficiencies.

TABLE II
SPEEDUPS AS A FUNCTION OF THE NUMBER OF THREADS AND THE SIZE OF THE IMAGE SET

Threads / Images	10	25	50	90	159
2	1.5358	1.5852	1.4635	1.5397	1.5677
4	2.1698	2.2162	1.9340	2.0703	2.1128
6	2.4740	2.3010	2.2087	2.3514	2.3303
8	2.7683	2.5189	2.5397	2.5028	2.5315

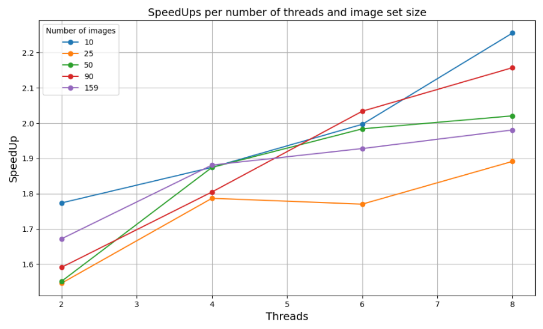

Fig. 4. Speedups

TABLE III
EFFICIENCY AS A FUNCTION OF THE NUMBER OF THREADS AND THE SIZE OF THE IMAGE SET

Threads / Images	10	25	50	90	159
2	0.8870	0.7732	0.7759	0.7957	0.8361
4	0.4687	0.4468	0.4686	0.4512	0.4703
6	0.3328	0.2951	0.3307	0.3390	0.3213
8	0.2820	0.2365	0.2526	0.2697	0.2476

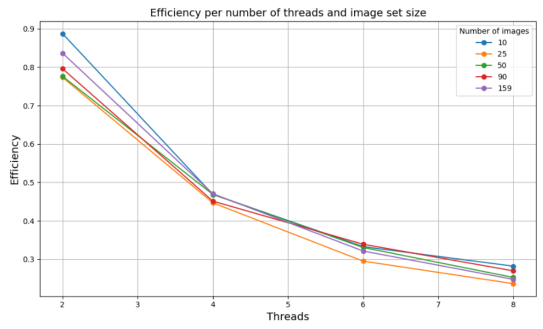

Fig. 5. Efficiencies

On the other hand, Figure 5 and Table III show the opposite trend observed in speedup, where efficiency significantly decreases as the number of threads increases. With 2 threads, efficiency is higher, around 88.69% for 10 images and approximately 83.61% for 159 images. However, with 8 threads, efficiency drops below 30% in all cases. Efficiency is consistently lower for larger image sets, regardless of the number of threads.

These findings underscore the importance of appropriately selecting the number of threads in relation to the volume of data to be processed to optimize both speedup and efficiency. Although speedups are higher with an increasing number of threads, there is no improvement in efficiency.

V. Conclusions

The study demonstrated that parallelizing the Fuzzy C-Means algorithm can significantly improve the processing time of image segmentation, achieving considerable speedups, especially with a moderate number of threads, while maintaining segmentation accuracy in each of the tests. However, a decrease in efficiency was observed with an increase in the number of threads.

The results have significant implications for real-time image processing, particularly in the medical field, where the ability to process large volumes of image data quickly and efficiently is crucial. The research demonstrated that parallelization is a viable strategy for enhancing the capability of image processing systems, facilitating faster diagnoses. Based on these findings, it is recommended to explore the dynamic optimization of the number of threads during runtime. This optimization would allow the number of threads used to be adjusted automatically according to the workload and the system's available resources, thereby maximizing the performance and efficiency of the processing.

Building on this work, more advanced approaches to the dynamic use of threads can be developed for implementation in future projects. Specifically, optimized libraries could be created for integration into medical software, enabling a complete and efficient implementation of the parallel FCM algorithm. These libraries would not only improve the speed and efficiency of real-time image processing but could also be adapted to other fields that benefit from such processing, such as surveillance, the automotive industry, and scientific research. Additionally, the dynamic optimization of the number of threads during runtime could become a standard feature of these systems, automatically adjusting according to the workload and the system's available resources, thereby maximizing processing performance and efficiency.

References

[1] H. Alsghaier, M. Akour, I. Shehabat, and S. Aldiabat, ”The importance of big data analytics in business: a case study,” American Journal of Software Engineering and Applications, vol. 6, no. 4, pp. 111-115, 2017.

[2] S. Rahimi, M. Zargham, A. Thakre, and D. Chhillar, ”A parallel fuzzy c-mean algorithm for image segmentation,” in Proc. IEEE Annual Meeting of the Fuzzy Information, NAFIPS'04., 2004, vol. 1, pp. 234-237.

[3] G. Dong and M. Xie, ”Color clustering and learning for image segmentation based on neural networks,” IEEE Transactions on Neural Networks, vol. 16, no. 4, pp. 925-936, 2005.

[4] V. Ganti, J. Gehrke, and R. Ramakrishnan, ”Mining very large databases,” Computer, vol. 32, no. 8, pp. 38-45, Aug. 1999.

[5] T. Chaira, ”A novel intuitionistic fuzzy C means clustering algorithm and its application to medical images,” Applied Soft Computing, vol. 11, no. 2, pp. 1711-1717, 2011.

[6] S. Zhou, D. Li, Z. Zhang, and R. Ping, ”A new membership scaling fuzzy c-means clustering algorithm,” IEEE Transactions on Fuzzy Systems, vol. 29, no. 9, pp. 2810-2818, 2020.

[7] S. J. Pennycook, J. D. Sewall, and J. R. Hammond, ”Evaluating the impact of proposed openmp 5.0 features on performance, portability and productivity,” in Proc. 2018 IEEE/ACM Int. Workshop on Performance, Portability and Productivity in HPC (P3HPC), Nov. 2018, pp. 37-46.

[8] T. Hulsen, ”Literature analysis of artificial intelligence in biomedicine,” Annals of Translational Medicine, vol. 10, no. 23, 2022.

[9] C. Demir and B. Yener, ”Automated cancer diagnosis based on histopathological images: a systematic survey,” Rensselaer Polytechnic Institute, Tech. Rep., 2005.

[10] T. M. Tuan, H. Fujita, N. Dey, A. S. Ashour, V. T. N. Ngoc, and D. T. Chu, ”Dental diagnosis from X-ray images: An expert system based on fuzzy computing,” Biomed. Signal Process. Control, vol. 39, pp. 64-73, 2018.

[11] V. V. Vela-Rincón, D. Mújica-Vargas, and J. de Jesus Rubio, ”Parallel hesitant fuzzy C-means algorithm to image segmentation,” in Signal, Image and Video Processing, vol. 16, no. 1, pp. 73-81, Feb. 2022.

[12] B. P. Vikraman and J. Afthab, ”Fuzzy C-Means Approach Optimized using Raindrop Algorithm for Image Segmentation,” in *Proceedings of the Fourth International Conference on Advances in Computer Engineering and Communication Systems (ICACECS 2023)*, vol. 18, no. 1, pp. 34-44, Apr. 2023.

[13] J. Pérez-Ortega, C. D. Rey-Figueroa, S. S. Roblero-Aguilar, N. N. Almanza-Ortega, C. Zavala-Díaz, S. García-Paredes, and V. Landero-Nájera, ”POFCM: A Parallel Fuzzy Clustering Algorithm for Large Datasets,” Mathematics, vol. 11, no. 1920, Apr. 2023.

[14] J. Han, M. Kamber, and J. Pei, ”Data Mining: Concepts and Techniques”, Southeast Asia ed. San Francisco, CA, USA: Morgan Kaufmann, 2006.

[15] A. Gosain and S. Dahiya, ”Performance Analysis of Various Fuzzy Clustering Algorithms: A Review,” Procedia Computer Science, vol. 79, pp. 100-111, 2016. doi: 10.1016/j.procs.2016.03.014.

[16] R. Xu and D. Wunsch, ”Partitional Clustering,” in ”Clustering”, 1st ed. Hoboken, NJ, USA: John Wiley & Sons, 2008, pp. 63-110. doi: 10.1002/9780470382776.ch4.

[17] M. Trobec, B. Slivnik, P. Bulić, and B. Robič, ”Introduction to Parallel Computing: From Algorithms to Programming on State-of-the-Art Platforms”, Springer, 2018.

[18] Cornell Virtual Workshop, ”Introduction to Parallel Computing”, 2022. [Online]. Available: https://cvw.cac.cornell.edu/parallel.

[19] C. Sha, J. Hou, and H. Cui, ”A robust 2D Otsu's thresholding method in image segmentation,” Journal of Visual Communication and Image Representation, vol. 41, pp. 339-351, 2016. doi: https://doi.org/10.1016/j.jvcir.2016.10.013.

[20] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, ”Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression,” arXiv preprint arXiv:1902.09630, 2019.

[21] J. Blahiri, ”BCCD Dataset with Mask,” [Dataset]. Disponible: https://www.kaggle.com/datasets/jeetblahiri/bccd-dataset-with-mask/data