Journal of Information Hiding and Multimedia Signal Processing ©2023 ISSN 2073-4212
Ubiquitous International Volume **14**, Number **3**, September **2023**

An approach based on optimizing speed of the binary algorithm Otsu

Trung Nguyen Tu*

Faculty of Information Technology, Thuyloi University, 175 Tay Son, Hanoi, Vietnam
Email: trungnt@tlu.edu.vn

Received March 2023; revised June 2023

Abstract. *Binaryization plays an important role in digital image processing, mainly in computer vision applications. Threshold generation is an effective technique in the binary encoding process. The choice of threshold technique is very important in the process of binaryization. Some threshold algorithms have been studied and proposed to determine optimal threshold values such as Otsu or Sauvola. In this study, the paper presents an improvement of the Otsu threshold algorithm based on MapReduce model to Improved performance when working on large images. Experimental results show that the speed of the improved algorithm is much better than the Otsu algorithm on many different types of images.*
Keywords: Otsu, Binary image, Binarization, Threshold generation, MapReduce.

1. **Introduction.** Binary images are useful in many image processing applications due to their simplicity and efficiency. Binary images are created by quantizing the image's grayscale levels into two values, usually 0 and 1. For many years, binary coding has been an active field of research in the field of processing. digital images such as recognizing text and symbols, identifying objects with special shadows and determining the orientation of objects... [3]. Binarization generally involves two steps including determination of a gray threshold according to some objective criteria and assigning each pixel to one class of background or foreground. If the pixels intensity is greater than the determined threshold then it belongs to foreground class and otherwise to the background [1]. The main problem in the binaryization process is the choice of threshold creation technique [2]. Several threshold algorithms have been proposed to determine the optimal threshold value. Threshold algorithms can be classified into six classes: histogram shape-based methods, clustering-based methods, entropy based methods, object attribute-based methods, the spatial methods and local methods based on the local characteristics of each pixel [6]. Among these classes, many threshold algorithms are based on a minimum variance [4]. The Otsu threshold technique is a classification-based method to find the threshold for minimizing internal variance, defined as the sum of the weighted variance of two classes [5]. The Gaussian Otsu method is an extension of the Otsu method and it uses the maximum variance between classes as the optimal threshold value. In [10], the authors experienced and compared two binary algorithms Otsu and Gaussian Otsu. In [11], Sauvola et al. proposed a locally adapted document binaryization algorithm, called the Sauvola binary algorithm. The Isauvola algorithm is an improvement of the Sauvola [12] algorithm. In [13][14], Romen et al. proposed a locally adaptive binary algorithm based on the integral image whose runtime does not depend on the local window size to compute the average in the local window.

However, the Otsu algorithm is still time-consuming when working with large images. Current imaging equipments is created towards the creation of high-resolution and large-scale images. The current remote sensing satellites also produce large images... Therefore, if the execution time of Otsu algorithm can be improved on large images, it will meet the new demand with big data...

Currently, with the development of information technology, the Industrial Revolution 4.0 has led to the explosion of data (Big Data). Big data and its analysis play an important role in the Information Technology world with applications of Cloud Technology, Data Mining, Hadoop and MapReduce [7]. Traditional technologies only apply to structured data while big data includes both structured, semi-structured and unstructured data. His method to effectively handle big data has become big challenges in the new age and there's a great need for new processing methods. MapReduce is a highly efficient distributed data processing model that has been widely used in large data processing [8]. In [9], the authors presented a parallel algorithm for boundary detection in images based on the Otsu-Candy operator. In [15], authors presented an method of optimizing Cross-Entropy Thresholding for image segmentation. In [16], author used the improved Swarm Algorithm to find the optimal thresholds for segmenting Medical images. In [17], authors proposed the Design of Real-time Face Position Tracking and Gesture Recognition System.

This paper presents the new binary algorithm MR_Fast_Otsu with the using MapReduce model to overcome the disadvantage of computation time of Otsu algorithm with large size image like remote sensing images without losing binary quality. The MR_Fast_Otsu algorithm promises to effectively support binary large images, saving a lot of execution time in the current data explosion.

The rest of the paper are presented as follows: Part II presents related studies, Part III presents proposed algorithms, Part IV presents some experiments on large image data. Finally, the conclusion is presented.

2. Related Work.

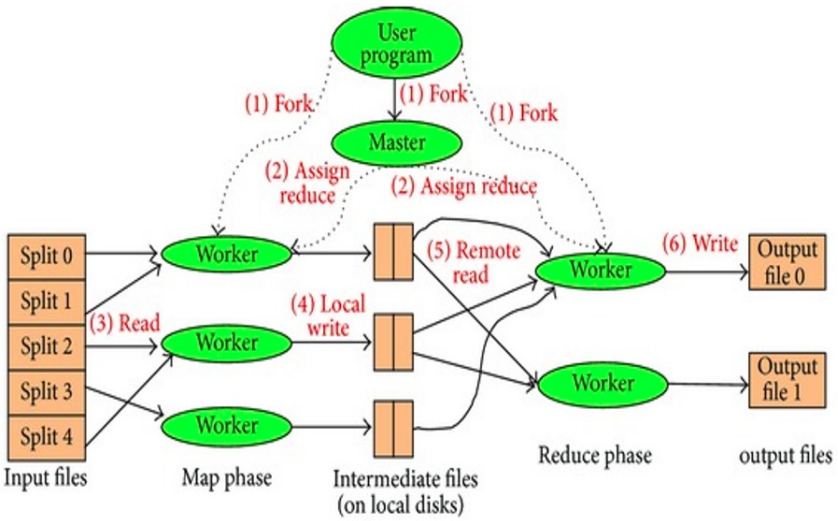

FIGURE 1. Flowchart of MapReduce model [8].

2.1. **Overview of MapReduce model.** MapReduce is a model of parallel and distributed computing model that is proposed by google (Figure 2). It includes two basic functions: "Map" and "Reduce" which are defined by the user [8]. Through the MapReduce library, the program fragments the input data file. Machines include: master and

worker. The master machine coordinates the operation of the MapReduce implementation process on the worker machines, the worker machines perform the Map and Reduce tasks with the data it receives. Data is structured in the form of key and value.

The formal representation of MapReduce model

According to [8] [13], we have the formal representation of the MapReduce model as follows:

- map: (K1 k1, V1 v1) → list(K2 k2, V2 v2)
- reduce: (K2 k2, list(V2 v2)) → list(K3 k3, V3 v3)

Where:

- K1, V1 are the input key and value types of the map function; k1, v1 are the corresponding objects with the types K1, V1.
- K2, V2 are the output key and value types of map function and still are the input key and value types of reduce function; k2, v2 are the the corresponding objects with the types K2, V2.
- K3, V3 are the output key and value types of the reduce function; k3, v3 are the the corresponding objects with the types K3, V3.

In other words, we can see:

- If k1, v1, k2, v2 are identified, we have the input and output of map function. Commonly, with text data, k1 is offset value of a data row, v1 is the content of a data row.
- If k2, v2, k3, v3 are identified, we have the input, and output of reduce function.

The formal Representation may be rewritten only with k1, v1, k2, v2, k3, v3 as follows:

$$map : (k1, v1) \rightarrow list(k2, v2) \tag{1}$$

$$reduce : (k2, list(v2)) \rightarrow list(k3, v3) \tag{2}$$

Figure 2 illustrates the diagram of the MapReduce job execution and data conversion from types (K1, V1) to types (K2, V2) and types (K2, V2) to types (K2, V3).

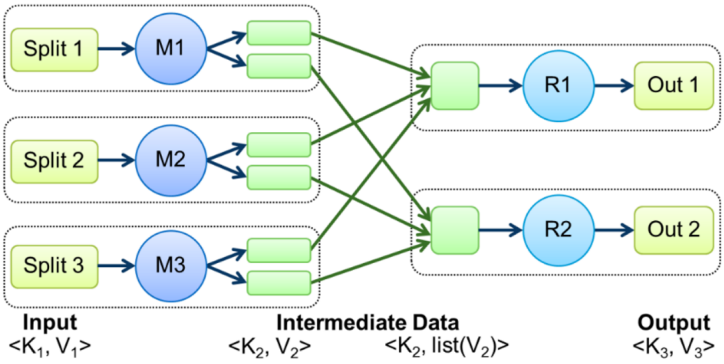

FIGURE 2. Flowchart of MapReduce model [12].

2.2. **The binary algorithm Otsu.** Otsu is the name of a Japanese researcher [5] who came up with the idea of calculating threshold automatically based on pixel value instead of using fixed threshold.

- Step 1: Calculating Histogram of the input image:

$$p_i = \sum_{i=0}^{L-1} \frac{n_i}{M * N} \tag{3}$$

Where:
- n_i: the number of pixels whole gray level is i.
- : L = 256.
- $p_0 + p_1 + ... + p_{L-1} = 1$

- Step 2: With every threshold $T_k = k$, $(0 < k < L - 1)$ to divide the input image into 2 classes C_1 (a set of pixels whole gray level is smaller than k) and C_2 (a set of pixels whole gray level is bigger than k)
  - Step 2.1: Calculating $P_1(k)$ and $P_2(k)$ corresponding is the ratio of the number of pixels in the class C_1 and C_2 that compared with the total number of pixels:

$$P_1(k) = \sum_{i=0}^{k} p_i \tag{4}$$

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i \tag{5}$$

  - Step 2.2: Calculating m_1, m_2 that are the mean values of the classes C_1, C_2:

$$m_1(k) = \sum_{i=1}^{k} iP(\frac{1}{C_1}) = \frac{1}{P_1(k)ip_i} \tag{6}$$

$$m_2(k) = \sum_{i=k+1}^{L-1} iP(\frac{1}{C_2}) = \frac{1}{P_2(k)ip_i} \tag{7}$$

  - Step 2.3: Calculating σ_B is the variance of the two classes C_1 and C_2:

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 \tag{8}$$

$$\sigma_B^2 = P_1P_2(m_1 - m_2)^2 \tag{9}$$

    Where:
    - m_G: mean value of the image:

$$m_G = \sum^{L-1} i = 0ip_i \tag{10}$$

      hay $m_G = P_1m_1 + P_2m_2$
    - m_k: the mean value to the threshold k:

$$m_k = \sum^{k} i = 0ip_i \tag{11}$$

- Step 3: According to Otsu, k* is calculated that the value of $\sigma_B^2(k^*)$, the difference between the two paragraphs (background color and character color), reaches the maximum value:

$$\sigma_B^2(k^*) = max_{0<=k<=L-1}\sigma_B^2(k) \tag{12}$$

  Note: If having the biggest values σ_B^2 that they are equal, k* is the biggest value k whole σ_B^2 is max.
- Step 4: Perform the image binarization according to threshold k*:

$$g_{Out}(x,y) = \begin{cases} 1 & if \\ g_{In}(x,y) > k* & \\ 0 & if \\ g_{In}(x,y) < k* & \end{cases} \tag{13}$$

Where:

– $g_{In}(x, y)$: value of input pixel
– $g_{Out}(x, y)$: value of output pixel

3. Propose the binary algorithm MR_Fast_Otsu.

3.1. **Analyze limitations of the binary algorithm Otsu.** Although the Otsu algorithm is very efficient in binary image, it still has limitations, for example: The execution speed is very slow with large images. As presented in Section 2.2, the computation time of this algorithm focuses mainly on the steps:

- Calculating Histogram: Browsing all the pixels to calculate the amount of each grayscale.
- Calculating the quantities $P_1(k), P_2(k), m_1(k), m_2(k)$ follow as the fomulas (6), (7), (8), (9)

3.2. **The algorithm MR_Fast_Otsu.** In this section, the author presents an improvement of the Otsu binary algorithm that is based on the MapReduce model and a quick solution of the quantities $P_1(k), P_2(k), m_1(k), m_2(k)$, called MapReduce_Fast_Otsu (MFO). The MFO algorithm still consists of 4 steps that are the same as the Otsu algorithm but making improvements in steps 1 and 2 corresponding to the calculation of Histogram and the quantities $P_1(k), P_2(k), m_1(k), m_2(k)$.

3.2.1. *Calculating Histogram using the MapReduce model.* In this section, the author proposes algorithm MR_Hist (MapReduce Histogram) to fix the problem of time when calculating the frequency of the Histogram level. Figure 3 is the diagram of the gray level calculation algorithm MR_Hist:

According to the algorithm flowchart, the input image is first converted to a list of gray levels of pixels suitable for MapReduce processing. Next, the system will divide the data into input splits, each processed in parallel by MapTask (which does a 1 count for the gray level value for the pixel being considered) to obtain the intermediate data. After all input splits are processed by MapTask, the intermediate data will be sorted, mixed, and grouped by gray level. Next, the grouped data will be processed by ReduceTask to calculate the total number of occurrences corresponding to the gray level in k2. The algorithm ends when all ReduceTasks are completed.

- Formal representation for procedures map_Hist and reduce_Hist Input: Each data element is a gray level value g_{ij} (of each pixel at the position of row i and column j). Output: A couple: Gray level values and the corresponding frequency of occurrence (g, count(g)). Then, we determine the pairs k1, v1 and k3, v3 as follows:
  – k1 is null or offset, v1 is g_{ij}.
  – k3 is the gray level value g, v3 is count(g). The Map function performs counting 1 for each gray level stored in v1 and so k2, v2 are inferred as follows:
  – k2 is g_{ij}, v2 is 1 At that moment, it has the form representation of the Map and Reduce procedures as follows:

$$map_Hist = (offset, g_{ij}) \rightarrow list(g_{ij}, 1) \tag{14}$$

$$reduce_Hist = (g, list(1)) \rightarrow list(g, count(list(1))) \tag{15}$$

- The procedure map_Hist Table 1 describes the algorithm of the procedure map_Hist. The purpose of the map_Hist algorithm is to count 1 for the occurrence of the gray level to be considered.
- The procedure reduce_Hist Table 2 describes the algorithm of the procedure reduce_Hist. The purpose of the algorithm reduce_Hist is to sum the numbers 1 which grouped by each grayscale.

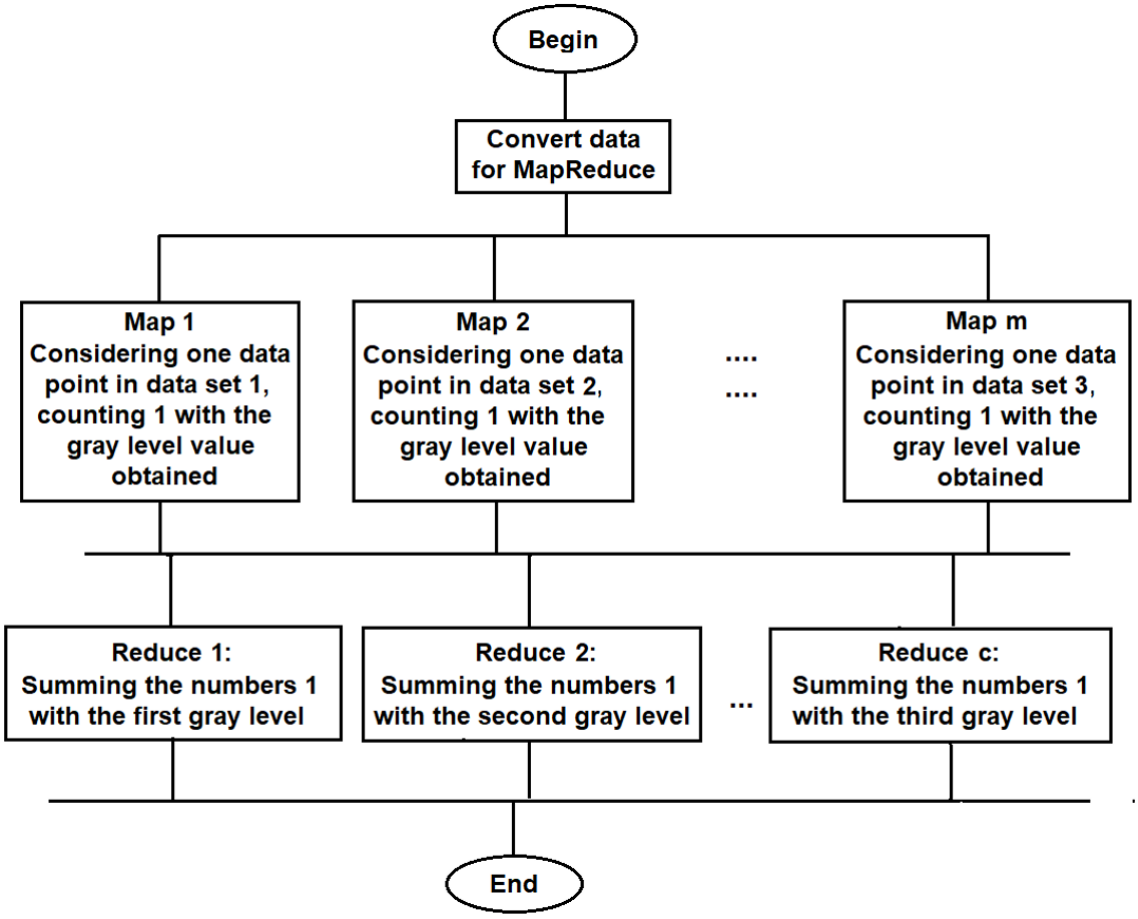

FIGURE 3. The flowchart of the algorithm MR_Hist.

TABLE 1. The algorithm of function map_Hist.

Input: key k1 is the values offset or null, value v1 is gray level g_{ij} Output: lstK2V2 is list of couples (k2,v2): k2 is g_{ij}, v2 is 1
Step 1: Assigning k2 = g_{ij} Step 2: Assigning v2 = 11 Step 3: Add the couple (k2,v2) into lstK2V2 Step 4: return lstK2V2

3.2.2. *Quick calculation of Coefficients* $P_1(k), P_2(k), m_1(k), m_2(k)$. According to fomulas (6) and (7) it could deduce as follows:

$$P_1(0) = hist(0) \tag{16}$$

$$P_1(k) = P_1(k-1) + hist(k), k > 0 \tag{17}$$

$$P_2(k) = totalPixel - P_1(k) \tag{18}$$

Where, totalPixel is the total number of pixels in the input image.
Attention: If it's considered:

$$m_1 = \sum_{i=0}^{k} iP_i \tag{19}$$

TABLE 2. The algorithm of function reduce_Hist.

Input: key is the value of gray level g, value is a list of values of 1 corresponding to the number of occurrences of the gray level g that is considered in the procedure map, it means list(1) Output: A couple (k3,v3): k3 is the gray level g, v3 is the sum of the elements in list(1)
Step 1: Initializing the variable sum = 0 Step 2: Assigning k3 = g Step 3: Foreach list(1) Step 3.1: Increasing sum = sum + 1 Step 4: Assigning k3 = sum

$$m_2 = \sum_{i=k+1}^{L-1} iP_i \tag{20}$$

then:

$$\sigma_B^2 = P_1P_2(\frac{m_1}{P_1} - \frac{m_2}{P_2})^2 = P_1P_2(\frac{m_1P_2 - m_2P_1}{P_1P_2})^2 = \frac{(m_1P_2 - m_2P_1)^2}{P_1P_2} \tag{21}$$

It's also deduced:

$$m_1(0) = 0 * hist(0) = 0 \tag{22}$$

$$m_1(k) = m_1(k-1) + k * hist(k), withk > 0 \tag{23}$$

$$m_2(255) = 0 \tag{24}$$

$$m_2(k) = m_1(k+1) + (k+1) * hist(k+1), k < 255 \tag{25}$$

With the above argument, it is easy to see that:

- $P_1(k)$ can be calculated based on $P_1(k-1)$ according to the fomula (16) or (18) instead of having to recalculate the whole thing like the formula (6) and it should save a lot of time.
- $P_2(k)$ can be calculated based on $P_1(k)$ according to the fomula (18) instead of having to recalculate the whole thing like the formula (7) and it should save a lot of time.
- Similar to $m_1(k)$, $m_2(k)$ can be calculated based on the fomulas (22-25) and it should save a lot of time.

4. **Experiment.** The author tested the proposed binary algorithm MFO and compared with binary algorithm Otsu.

Experimental data includes 3 types of images: Remote sensing (RS), Document (Doc) and Landscape (LS) images. Due to the limitation of the paper, the author demonstrates the experiment with different input image samples as shown in figure 4a, 4b, 4c and Table 3. In this experiment, i use the tool Spark to implement the algorithm MFO.

TABLE 3. Some representative samples and experiencing results.

width	height	Otsu Time	MFO Time	Otsu/MFO
7173	7202	13236	1395	9.49
4456	4465	5506	633	8.70
2479	3508	4722	509	9.28
2550	2300	4392	611	7.19
3264	2448	1205	527	2.29
2448	3264	2784	474	5.87

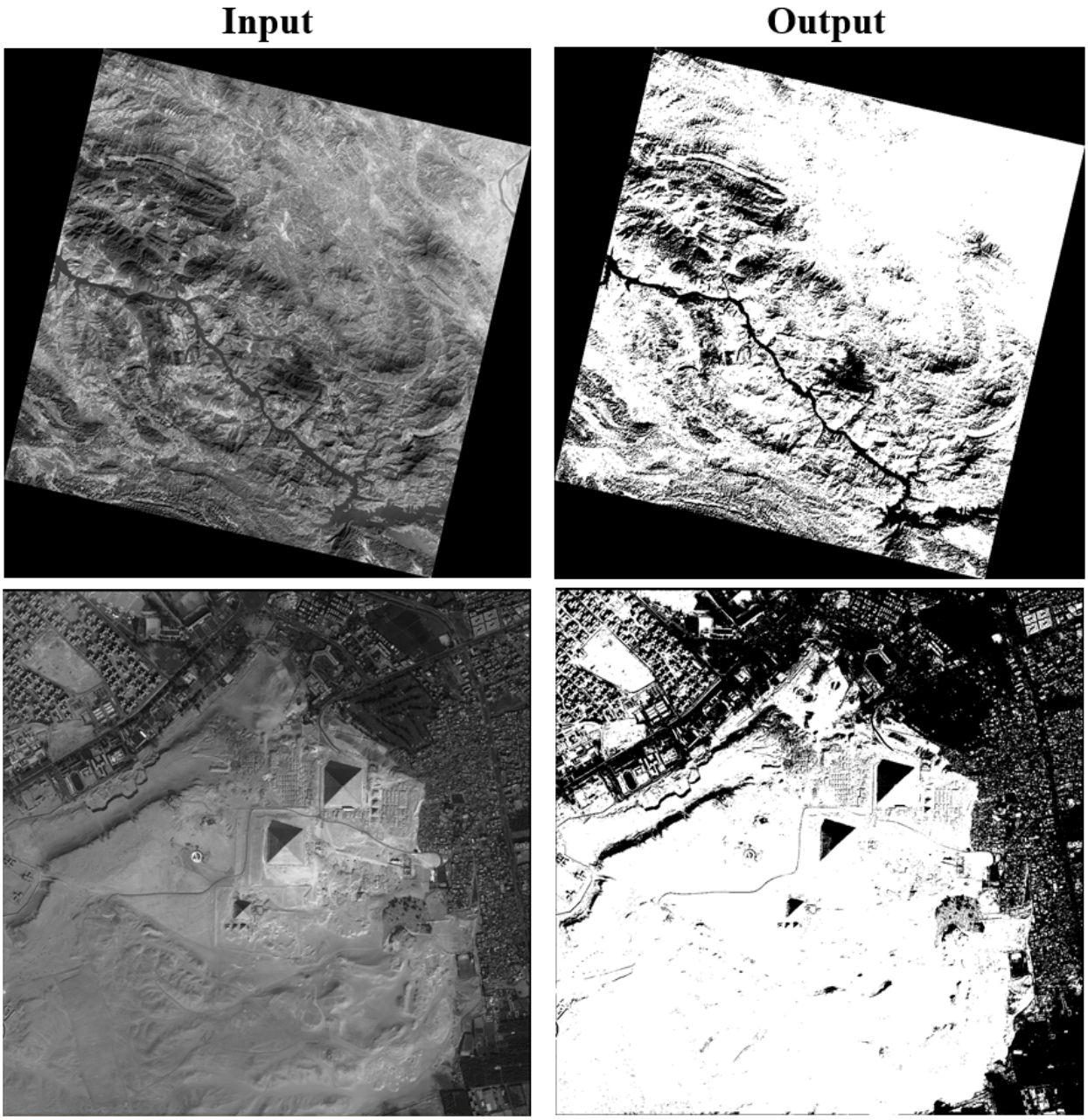

FIGURE 4. Illustrating some image binary results with remote sensing images.

Averaging times of Otsu and MFO are 5307.5 691.5 correspondingly. And Otsu/MFO time is 7.68.

Comments: From the above table is shown that:

- The largest time difference between Otsu and MFO is 9.49.
- The largest time difference between Otsu and MFO is 2.29.
- In the most tests, the time difference between Otsu and MFO is about 6 times or more.
- The average time difference between Otsu and MFO is 7.68
- The execution time of the algorithm MFO is much smaller than the original algorithm Otsu.

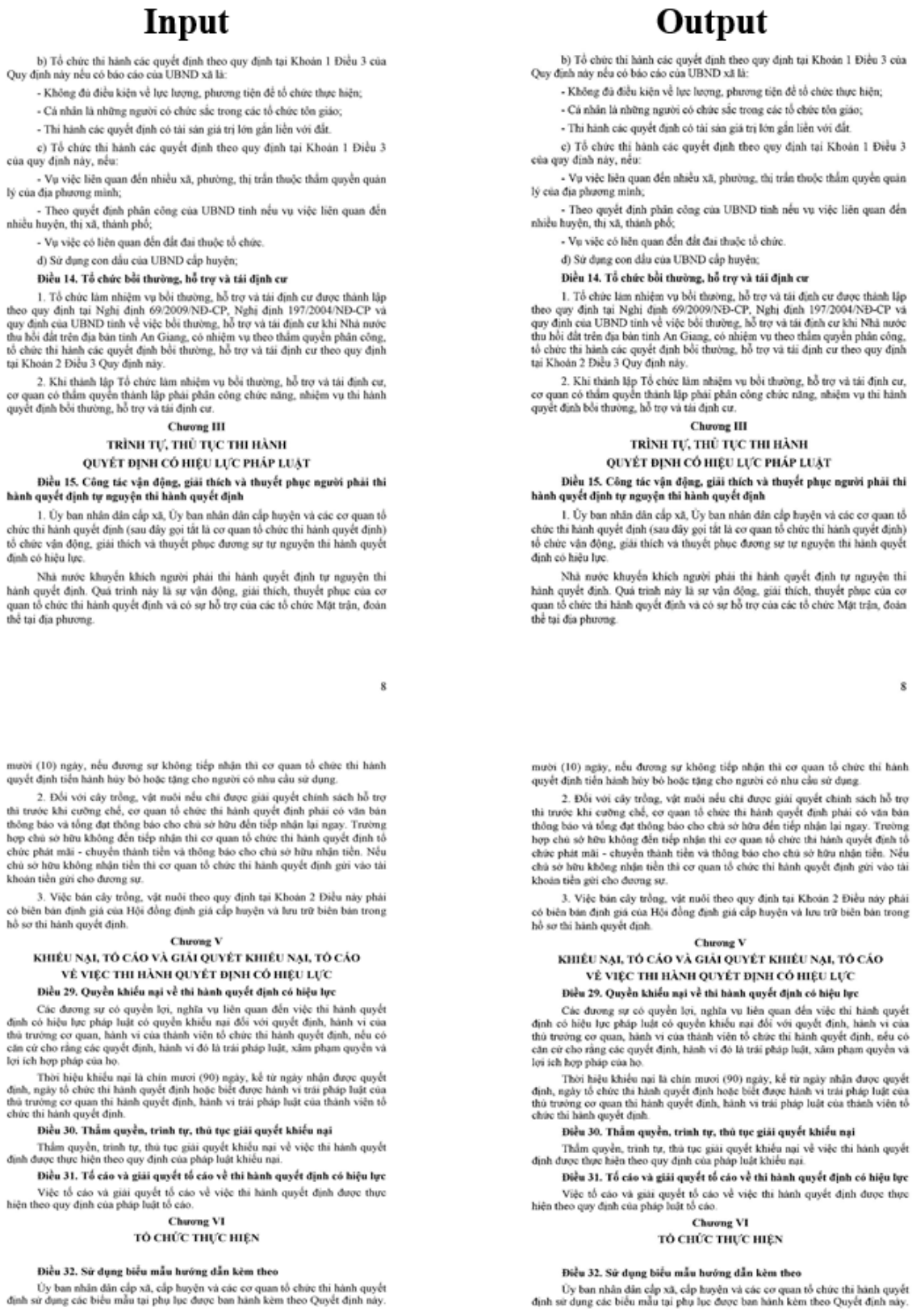

FIGURE 5. Illustrating some image binary results with document images.

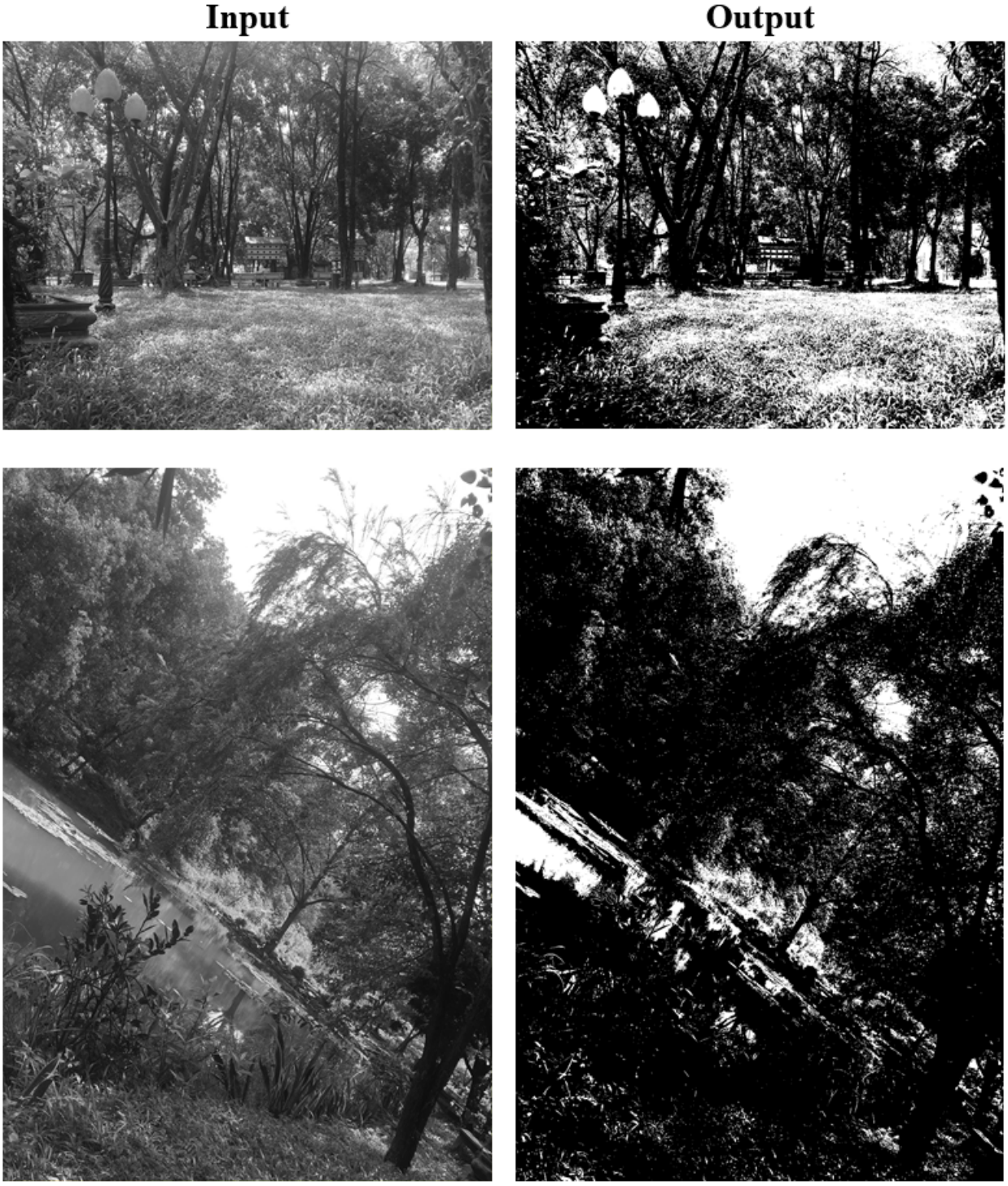

FIGURE 6. Illustrating some image binary results with landscape images.

5. **Conclusions.** In this paper, the author proposed an improvement of the Otsu binary algorithm for large images. The proposed algorithm uses the MapReduce model and a fast computation solution for the key components of the Otsu algorithm. The test results on three different types of images showed an average increase of about 7.7 times the execution speed compared to the original algorithm. In the next research, the author will continue to study binary algorithms and image classification.

Acknowledgment. This work is partially supported by Department of Information Technology, Thuyloi university and Institute of Information Technology, Vietnamese Academy of Science and Technology. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] Jain B. D, "Goal directed evaluation of binarization methods", IEEE Transactions on Pattern Analysis and MAchine Intelligence, 17:11911200, 1995.
[2] Melgani F, "Robust image binarization with ensembles of thresholding algorithms", J. Electron. Imaging, 15, 2006.
[3] Luck Fletcher, "Emg tutorial Technical report", Available at http://users.cecs.anu.edu.au/ Luke.
[4] Z. H. Nowinsk, "On minimum variance thresholding", Journal of Pattern Recognition Letters, 15:17321743, 2006.
[5] N. Otsu, "A threshold selection method from gray level histograms", IEEE Trans. systems. Man. and Cybernetics, VOL. SMC-9, NO. 1, JANUARY 1979.
[6] B. S. Sezgin, "A survey over image thresholding techniques and quantitative performance evaluation", Journal of Electronic Imaging, 13:146165, 2004.
[7] Nandhini.P, "A Research on Big Data Analytics Security and Privacy in Cloud, Data Mining, Hadoop and Mapreduce", Int. Journal of Engineering Research and Application, 2018.
[8] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", USENIX Association OSDI '04: 6th Symposium on Operating Systems Design and Implementation, 2004.
[9] Jianfang Cao et al, "Implementing a Parallel Image Edge Detection Algorithm Based on the Otsu-Canny Operator on the Hadoop Platform", Computational Intelligence and Neuroscience, May 2018.
[10] Yousefi, Jamileh, "Image Binarization using Otsu Thresholding Algorithm", 2015. 10.13140/RG.2.1.4758.9284.
[11] Sauvola, J., Seppanen, T., Haapakoski, S., and Pietikainen, M.: "Adaptive document binarization". Proc. 4th Int. Conf. on Document Analysis and Recognition, Ulm Germany, 1997, pp. 147–152
[12] Zineb H. et al, "ISauvola: Improved Sauvola's Algorithm for Document Image Binarization", ICIAR 2016, LNCS 9730, pp. 737–745, 2016, DOI: 10.1007/978-3-319-41501-$7_8$2.
[13] Tom White, "Hadoop: The Definitive Guide : The Definitive Guide", 2009.
[14] T.Romen Singh et. al, "A New Local Adaptive Thresholding Technique in Binarization", International Journal of Computer Science Issues, Vol. 8, Issue 6, No 2, November 2011.
[15] Jeng-Shyang Pan, Thi-Lan-Phuong Nguyen, Truong-Giang Ngo, Thi-Kien Dao, Thi-Thanh-Tan Nguyen, Trong-The Nguyen, "An Optimizing Cross-Entropy Thresholding for Image Segmentation based on Improved Cockroach Colony Optimization", Journal of Information Hiding and Multimedia Signal Processing, Vol. 11, No. 4, pp. 162-171, December 2020.
[16] Trong-The Nguyen, Trinh Dong-Nguyen, Truong-Giang Ngo and Vinh-Tiep Nguyen, "An Optimal Thresholds for Segmenting Medical Images Using Improved Swarm Algorithm", Journal of Information Hiding and Multimedia Signal Processing, Vol. 13, No. 1, pp. 12-21, March 2022.
[17] Teh-Lu Liao, Han-Chang Chen and Jun-Juh Yan, "Design of Real-time Face Position Tracking and Gesture Recognition System based on Image Segmentation Algorithm", Journal of Network Intelligence, Vol. 5, No. 4, pp. 226-239, November 2020.