ISSN 1840-4855
e-ISSN 2233-0046

Original scientific article
http://dx.doi.org/10.70102/afts.2025.1834.698

OTSU AND KAPUR ENTROPY BASED OPTIMAL MULTILEVEL IMAGE THRESHOLDING USING JAYA AND STOCHASTIC FRACTAL SEARCH ALGORITHMS FOR ENHANCED IMAGE SEGMENTATION

S. Anbazhagan[1], M. Karthika[2*], S. Ramkumar[3], P. Nammalvar[4], P. Anbarasan[5], V. Krishnakumar[6]

[1]Associate Professor, Faculty of Engineering and Technology, Annamalai University, Annamalai Nagar, Tamil Nadu, India. e-mail: s.anbazhagan@gmx.com, orcid: https://orcid.org/0000-0001-7167-4414
[2]Associate Professor, Department of Electrical and Electronics Engineering, New Horizon College of Engineering, Bengaluru, Karnataka, India. e-mail: karthikaganesh16@gmail.com, orcid: https://orcid.org/0000-0002-2832-1426*
[3]Professor, Department of Electrical and Electronics Engineering, Kangeyam Institute of Technology, Tiruppur, Tamil Nadu, India. e-mail: set.dean@gmail.com, orcid: https://orcid.org/0000-0001-7426-5034
[4]Associate Professor, Department of Electrical and Electronics Engineering, Kangeyam Institute of Technology, Tiruppur, Tamil Nadu, India. e-mail: alvar1976@gmail.com, orcid: https://orcid.org/0000-0002-4448-9333
[5]Associate Professor, Department of Electrical and Electronics Engineering, St. Joseph's Institute of Technology, OMR, Chennai, Tamil Nadu, India. e-mail: p.anbarasan@gmail.com, orcid: https://orcid.org/0000-0003-3182-1621
[6]Associate Professor, Department of Electrical and Electronics Engineering, St.Joseph's College of Engineering, OMR, Chennai, Tamil Nadu, India. e-mail: krishnakumarv@stjosephs.ac.in, orcid: https://orcid.org/0000-0001-6319-5237

Received: September 16, 2025; Revised: October 27, 2025; Accepted: December 01, 2025; Published: December 30, 2025

SUMMARY

Image segmentation plays an important role in medical diagnosis and recognition, but the traditional methods of multilevel thresholding have exponential computation complexity with the number of thresholds. The study corresponds to the necessity to have a computationally effective parameter-free optimization to support fast clinical decision-making. The study suggests two optimization systems that are used to optimize image segmentation, namely the Jaya algorithm and Stochastic Fractal Search (SFS). Jaya algorithm, with its single-phase update mechanism and no algorithm-specific parameters, is used to calculate optimal thresholds based on the maximization of Entropy in Kapur. At the same time, the SFS algorithm is based on the idea of natural fractal patterns and the diffusion of particles, which are used to maximize the between-class variance of Otsu. These two techniques were strictly tested on 256 × 256 8-bit benchmark images (Cameraman, Lena, and Peppers). The results of numerical assessments indicate that the two algorithms are able to approach optimal threshold values irrespective of varying levels (K = 2, 3, 4, 5). In the Jaya algorithm, especially, the computational efficiency was much better, and the minimum processing time was used without compromising the quality of segmentation. When compared to the existing metaheuristics such as GA and PSO, it is shown that the suggested methods are more stable

and robust and do not change in levels even when the number of thresholds grows. The results place the Jaya and SFS algorithms as promising algorithms to perform multi-level thresholding of images. They are very appropriate in real-time medical imaging and other technical applications that demand high-quality segments with minimal computational overhead.

Key words: *image segmentation, kapur's entropy, multilevel thresholding, soft computing, optimal thresholds, jaya algorithm, optimization techniques.*

INTRODUCTION

A popular method of image segmentation is thresholding. The key problem of thresholding is to select a suitable threshold value to divide the spatial domain of the image into significant parts. The method is important in a number of image processing cases, including machine-generated or handwritten text recognition and object shape identification, as well as image improvements. The underlying idea of image thresholding is to identify a single threshold (in bi-level thresholding) or more than one threshold (in multilevel thresholding) of an image with the intent of labelling pixels into specific regions [1]. In recent years, the increasing complexity of digital images, characterized by variations in intensity and homogeneity, has brought multilevel thresholding (MT) techniques into greater focus. This heightened attention is primarily due to the method's straightforward implementation and its characteristic of demanding minimal storage memory [2].

Multilevel thresholding (MT) transforms the process of image thresholding into an optimization problem, where the appropriate thresholds are determined by either maximizing or minimizing a certain measure. A notable example is Otsu's technique [3], which maximizes the between-class variance to determine thresholds. Alternatively, in the context of Kapur's entropy [4], optimal thresholds are attained by increasing the entropy across different classes. Researchers have also devised additional optimal criteria, such as Boltzmann–Gibbs entropy [5], Rényi's entropy [6], and others. The utilization of the firefly algorithm (FA) has been implemented to enhance the efficiency of multilevel image thresholding. However, there are instances where the FA can become trapped in local optima. The fundamental concept behind the improved firefly algorithm (IFA) involves dynamically selecting a strategy to guide fireflies towards optima based on varying stagnation conditions. Additionally, the multilevel Otsu thresholding function is adopted as the objective function, and the IFA is employed for the exploration of multilevel thresholds [7]. The objective of techniques such as whale optimization, moth-flame optimization, and hyper-heuristic methods is to determine optimal thresholds that maximize Otsu's function. Experimental outcomes of the proposed approaches have been compared with various swarm methods [8] [9].

Among these strategies, the Kapur method stands out for its ability to select the optimal threshold value by maximizing the entropy across different classes, garnering significant attention from seasoned researchers. However, this approach does possess a clear drawback: its computational complexity increases exponentially as the number of required thresholds rises. To some extent, this limitation restricts its applicability in the context of multilevel thresholding. As a result, various approaches and corresponding enhancements have been introduced to overcome the aforementioned drawbacks. The objective of the whale optimization and moth-flame optimization techniques is to determine optimal thresholds that maximize the Kapur function. Experimental results of the suggested approaches have been compared with a range of swarm methods [8]. The method put forward in this study is evaluated using standard test images and is compared against bacterial foraging, modified bacterial foraging, particle swarm optimization (PSO), genetic algorithm (GA), as well as a hybrid approach termed PSO-differential evolutionary. To tackle the multilevel thresholding (MT) image thresholding challenge with the Kapur entropy strategy, the firefly algorithm (FA) is employed. The effectiveness of the suggested method is assessed using standard test images, and the utilization of Levy flight demonstrates effective exploration capabilities [10].

A novel algorithm called Symbiotic Organisms Search (SOS) has been developed to improve multi-thresholding using Kapur's objective function. It's compared to six other algorithms, including PSO, FA, ABC, GA, and GWO [11][12]. The SFS algorithm stands out as a powerful approach for image

thresholding relying on Otsu's technique. It's tested on three standard datasets using fitness values and the Jaccard measure.

On the other hand, the Jaya algorithm is designed for image segmentation using Kapur's technique and doesn't require specific parameters. It refines outcomes using the Kapur entropy function iteratively until a termination condition is met. Its performance is compared to other metaheuristic algorithms on standard test images using fitness values and the Jaccard measure. The subsequent sections of this study are structured as follows: Section 2 introduces the problem formulation and outlines the significance of the Otsu and Kapur methods. Section 3 elaborates on the proposed techniques for multilevel thresholding based on SFS and the Jaya algorithm, along with detailing the methodology's implementation. Section 4 provides the analysis of results. Finally, the concluding section encapsulates the findings and outlines future research directions.

Current literature identifies image thresholding as one of the foundations of segmentation, which has developed over the years since the simplistic bi-level tasks to its multilevel form of thresholding (MT). Although basic requirements such as the between-class variance as proposed by Otsu and Entropy by Kapur are mathematically sound, the survey points to an important curse of dimensionality, with the increase in calculations being multiplied exponentially with new thresholds.

In order to counteract this, scholars have shifted their attention to metaheuristic algorithms, i.e., Firefly (FA), Particle Swarm (PSO), and Genetic Algorithms (GA). Nevertheless, most of the current approaches are susceptible to the issue of local optima traps or need fine-tuning of the parameters. This study fills these gaps by providing the Jaya and Stochastic Fractal Search (SFS) algorithms. In contrast to the previously used swarm-based algorithms, Jaya is characterized by a parameter-free simplicity that decreases the computation costs, and SFS is better at exploring using fractal-based diffusion. This work improves the advantages of segmentation and processing speed by applying these to Kapur and Otsu functions to overcome the drawbacks of conventional optimization literature.

MULTILEVEL THRESHOLDING CRITERION

Multilevel thresholding criterion refers to the specific set of rules, conditions, or mathematical measures used to determine optimal threshold values for the segmentation of an image into multiple levels or classes. This criterion aims to find the thresholds that maximize certain properties or measures, such as between-class variance, entropy, or other relevant indicators, in order to achieve effective and meaningful image segmentation. The various multilevel thresholding techniques can use different criteria in determining the optimum thresholds to distinguish different regions or classes in the image. This occurs by optimizing a target function, for which the chosen thresholds are demonstrated to be the parameters. The use of the Otsu technique for thresholding is a nonparametric method, which is designed to subdivide the whole picture into different areas in order to maximize the distribution or variation of different classes. As Otsu's technique is well-established, an in-depth discussion of it is not presented here. Interested readers can refer to [3][7][8][9] for further details. Similarly, thresholding employing the Kapur technique also follows a nonparametric strategy, leading to the partitioning of the complete image into several regions. This aims to optimize the entropy to its maximum potential and statistical spreading of pixel values in the image histogram. As the Kapur technique is widely acknowledged, an exhaustive elaboration on it is not included in this context. Individuals interested in the in-depth details can refer to [4][7][10][11] to find the information.

Otsu’s Technique

The technique devised by Otsu is a complicated approach that is used to decide on the appropriate threshold value that can be used to divide an image into two main categories, usually the foreground image and the background image. The concept behind it is that this segmentation is made through the manipulation of pixel luminance values. The essence of the Otsu method consists of the judicious control of the variance in and within these two classes. It aims at reducing the variance in each class, which is often referred to as intra-class variance, and at the same time increasing the difference between these two classes, and this is referred to as inter-class variance. In this way, the technique will attempt to

produce the clearest and most definite contrast between the image of the foreground and the background. The technique applied by Otsu is a systematic search of a variety of threshold values. At every threshold value, the algorithm determines the variances of the resulting foreground image and background image classes. These variances are basically a measure of the variation or the deviation of the pixel intensity in each category. The most important goal in this case is to find the threshold that would provide the greatest between-class variance or, in other words, the minimal within-class variance. Practically, what this entails is that the method used by Otsu is exhaustive and evaluates a large number of possible thresholds, calculating the associated variances. It then picks the threshold, which, when used, will result in the maximum difference between the classes of foreground and background image, forming the strongest of divides.

This is a well-selected value of threshold that is very important in image segmentation. It is an accurate delimiting force, which allows dividing the image into discrete and coherent regions or parts. This part of the segmentation is essential in any image processing system, including object recognition, image enhancement, and feature extraction, because it isolates the meaningful components of the image that can be further analyzed and manipulated. The technique presented by Otsu is very basic, but effective, and it has found extensive application in numerous fields, including image processing, object detection, and computer vision. It offers an automated method of setting threshold values without prior information as to the contents of the image, or manually setting threshold values. The non-parametric method of segmentation known as between-class variance [7] is where the maximization of between-class variance is carried out, and therefore reduces the within-class variance of the pixels belonging to a particular class. This calculation is usually done using the addition of the sigma functionality of every single class. The sigma operation of each of the classes is determined as follows in equation (1):

$$\sigma_0 = \omega_0(\mu_0 - \mu_T)^2 \quad \sigma_1 = \omega_1(\mu_1 - \mu_T)^2 \; \sigma_2 = \omega_2(\mu_2 - \mu_T)^2 \quad \sigma_k = \omega_k(\mu_k - \mu_T)^2 \tag{1}$$

where μ_0, μ_1, μ_2, and μ_k are the means of each class and can be calculated as in equation (2):

$$\mu_0 = \sum_{i=0}^{t1-1} \frac{ip_i}{\omega_i} \quad \mu_1 = \sum_{i=t1}^{t2-1} \frac{ip_i}{\omega_i} \quad \mu_2 = \sum_{i=t2}^{t3-1} \frac{ip_i}{\omega_i} \quad \mu_k = \sum_{i=tk}^{L-1} \frac{ip_i}{\omega_i} \tag{2}$$

and μ_T is the total mean value of the image, and it can be calculated as in equation (3):

$$\mu_T = \sum_{i=0}^{L-1} ip_i \tag{3}$$

The optimal thresholds are determined by maximizing the value of the cost function in equation (4):

$$f\left(\underset{t}{\rightarrow}\right) = \arg\, max\left(\sum_{i=0}^{k} \sigma_i\right) \tag{4}$$

The method of between-class variance based on Otsu has been applied to ascertain the finest threshold values for image segmentation. Furthermore, Otsu's method can be extended to accommodate multi-level thresholding. This extension assumes the presence of m thresholds, resulting in the separation of the image into m + 1 distinct classes.

Kapur Entropy

Kapur entropy is measured based on the probabilities of pixel intensities within different classes or segments obtained through thresholding. The measure takes into account both the inter-class variability (between-class entropy) and the intra-class variability (within-class entropy) to assess the quality of the resulting segmentation. In thresholding applications, the highest aim is to find threshold values that maximize the Kapur entropy. It implies that the resulting segments must possess as much uncertainty or randomness in their distribution of pixel intensities as possible, which implies that regions with distinct properties are effectively separated. Kapur entropy is a quantitative measure of the quality of image segmentations, which can be applied to different optimization algorithms to perform automatic determination of the optimal threshold values. It is among the techniques used in image processing to improve the accuracy and effectiveness of image processing tasks like object recognition, edge

detection, and feature extraction. Originally developed in bi-level thresholding, the entropy that Kapur developed can be extended to multilevel thresholding. When it comes to bi-level thresholding, only one threshold value is used, namely t, to partition the image into two distinct classes, namely C0 and C1, which is decided by the function in equation (5).

$$f_1(t) = H_0 + H_1 \tag{5}$$

where, $H_0 = -\sum_{i=0}^{t-1} \frac{p_i}{\omega_0} \ln \frac{p_i}{\omega_0}$, $\omega_0 = \sum_{i=0}^{t-1} p_i$ and $H_1 = -\sum_{i=t}^{L-1} \frac{p_i}{\omega_1} \ln \frac{p_i}{\omega_1}$, $\omega_1 = \sum_{i=t}^{L-1} p_i$ are the entropies. ω0 and ω1 are the class probabilities of the segmented classes. C0, C1, and the probability pi of each grey level i are calculated by following in equation (6).

$$p_i = \frac{h(i)}{\sum_{i=0}^{L-1} h(i)}, i = 0, \dots, L-1 \tag{6}$$

For multilevel thresholding, let there be m, the total number of thresholds [t0, t1, t2,…, tm] to be nominated, which separate the image into the multiple classes: C0, C1, C2,…, Cm by the function in equation (7),

$$f_1(t_1, t_2, \dots, t_m) = H_0 + H_1 + H_2 + \dots + H_m \tag{7}$$

Where, $H_0 = -\sum_{i=0}^{t_1-1} \frac{p_i}{\omega_0} \ln \frac{p_i}{\omega_0}$, $\omega_0 = \sum_{i=0}^{t_1-1} p_i$, $H_1 = -\sum_{i=t_1}^{t_2-1} \frac{p_i}{\omega_1} \ln \frac{p_i}{\omega_1}$, $\omega_1 = \sum_{i=t_1}^{t_2-1} p_i$, $H_2 = -\sum_{i=t_2}^{t_3-1} \frac{p_i}{\omega_2} \ln \frac{p_i}{\omega_2}$, $\omega_2 = \sum_{i=t_2}^{t_3-1} p_i$ and $H_m = -\sum_{i=t_m}^{L-1} \frac{p_i}{\omega_m} \ln \frac{p_i}{\omega_m}$, $\omega_m = \sum_{i=t_m}^{L-1} p_i$ are the entropies. ω0, ω1, ω2,…, ωm are the class probabilities of the segmented classes. C0, C1, C2,…, Cm.

PROPOSED METHODOLOGY

The study suggests a two-step methodology that combines two sophisticated metaheuristic optimization algorithms, Stochastic Fractal Search (SFS) and Jaya, in order to find the optimal thresholds for image segmentation. The system will solve the computational bottlenecks of classic multilevel thresholding with the use of parameter-free or low-parameter search mechanisms.

The process of methodology has a systematic way through which the input images are fed to the production of the segmented output. It starts with the calculation of the image histogram, which is the search space of the optimization algorithms. SFS or Jaya is applied to determine the best set of threshold values, $T_1, T_2, \dots T_k$, etc., depending on the objective function used (Otsu Variance or Kapur Entropy).

Figure 1 represents the suggested dual-pathway architecture of multilevel image segmentation. It starts with the Input Image, then moves on to the calculation of the histogram to examine the distribution of the pixels. It is followed by a bifurcation of the architecture to two different optimization streams, where the left stream applies the maximum variability of the Variance of Otsu by diffusion and update operations to the global exploration. At the same time, the right path optimizes the Jaya Algorithm to maximize the Entropy of Kapur, through an attraction-repulsion mechanism to optimize the best thresholds ($T_1, T_2, \dots T_k$). The two streams meet at the Multilevel Image Segmentation stage and eventually produce a precise output of a high quality, namely the Segmented Output Image, which could be used in technical analysis.

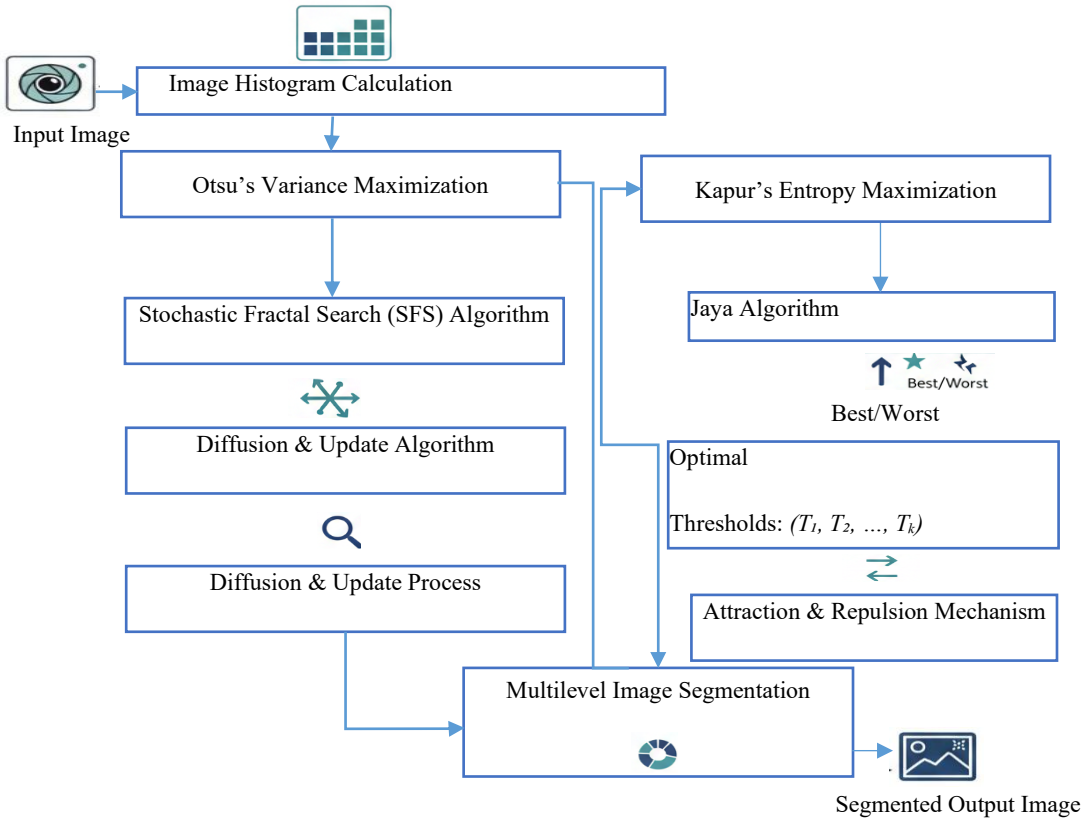

Figure 1. Integrated framework for multilevel thresholding using SFS and jaya algorithms

ALGORITHM FOR MULTILEVEL THRESHOLDING

Optimization refers to the process of enhancing a system beyond its previous state. Over the past decade, researchers have been captivated by the collective intelligent actions exhibited by groups of insects or animals in their natural habitats. Examples include the coordinated flight of birds, the organized movements of ant colonies, the synchronized swimming of fish schools, the coordinated efforts of bee swarms, and the collaborative behaviours of termites. This combined behaviour of insects, birds, or animals is recognized as swarm behaviour. Many experts have adopted swarm behaviour as a strategy for addressing intricate real-world challenges. Furthermore, all algorithms inspired by nature necessitate the adjustment of algorithmic parameters for effective performance. To circumvent this challenge, a parameter-free optimization technique known as the SFS and Jaya algorithms is currently implemented to tackle complex multi-dimensional problems.

SFS Algorithm

The SFS algorithm involves two main mechanisms, namely the diffusion process and the updating process. In the initial mechanism, reminiscent of Fractal Search, each particle undergoes diffusion around its current position to satisfy the escalation (exploitation) property. This step enhances the likelihood of discovering global minima and prevents entrapment within local minima. In the subsequent mechanism, SFS emulates how a point within the group adjusts its position based on the states of other points within the group. Diverging from the diffusing stage in the FS approach, which leads to a significant improvement in the number of involved points, SFS employs a static diffusion process. It means that in the SFS algorithm, only the best-performing particle, the one that fits the problem the best after exploring different possibilities, is kept. The rest of the particles are not taken into consideration. The randomness is also implemented in SFS in its steps of updating. This is randomness that is meant to assist in the exploration of the problem space. The updating process by SFS is created to balance between exploration of various alternatives (exploration) and optimization of the current best solution. For a deeper understanding of SFS and related research, interested researchers can refer to the following

sources: [12][13][14][15][16][17][18][19][20][21][22]. These references give additional information and insights into the SFS algorithm and its applications.

The implementation phases of the SFS algorithm can be outlined as follows [13] represent in equation (8), (9), (10), (11), (12), (13), (14):

1. Initialization:

$$P_i = LB + (UB - LB) \times \varepsilon \tag{8}$$

- Set algorithm parameters: Define the number of particles, maximum iterations, convergence criteria, and other control parameters.
- Initialize the population: Generate a set of particles randomly within the problem's feasible region.
- Evaluate fitness: Calculate the fitness value for each particle based on the objective function of the problem.

2. Diffusion Process:

$$GW_1 = Gaussian(\mu_{GBP}, \sigma) + (\varepsilon.BP - \varepsilon'.P_i) \tag{9}$$

For every element in the population:

- Generate a random displacement vector: This vector determines how much the particle will diffuse from its current position.
- Update the particle's position: Add the displacement vector to the particle's current position to obtain a new potential solution.
- Evaluate the appropriateness of the new position: Calculate the fitness value for the particle's new position.
- Update personal best: If the new fitness value is superior to the particle's personal best fitness, update the personal best position.

3. Select the Best Particle:

- Identify the particle with the finest fitness value among the entire population after the diffusion process.

4. Updating Process:

$$GW_2 = Gaussian(\mu_P, \sigma) \tag{10}$$

$$Pr_i = \frac{rank(P_i)}{N} \tag{11}$$

- For every element in the population:
- Update the particle's position to the best particle's position obtained from the diffusion process.
- Evaluate the fitness of the new position and update.

$$P_i'(j) = P_r(j) - \varepsilon \times \left(P_t(j) - P_i(j)\right)$$

$when Pr_i < \varepsilon$

$$No\ change\ when\ Pr_i \geq \varepsilon \tag{12}$$

$$P_i^{"} = P_i' - \tilde{\varepsilon} \times (P_t' - BP) |\ \varepsilon' \leq 0.5$$

$$P_i^{"} = P_i' + \tilde{\varepsilon} \times (P_t' - P_r') |\ \varepsilon' > 0.5 \tag{13}$$

5. Termination Criteria:

- Check convergence: Monitor whether the algorithm has converged based on predefined criteria, such as the highest number of iterations or a small change in the best fitness value.
- If convergence is reached or the maximum iterations are exceeded, proceed to the subsequent step.

6. Solution Retrieval:

- Return the position of the particle with the best fitness value as the optimized result to the multi-dimensional problem.

7. Wrap-up:

- Analyse and interpret results: Evaluate the obtained solution, analyse convergence behaviour, and relate it to other optimization approaches if applicable.
- Fine-tuning: Adjust algorithm parameters if necessary to achieve better performance on similar problems.

Where UB and LB represent the greater and minor boundaries for each dimension, ε and ε' are the arbitrary numbers produced by a normal spreading within the range [0, 1], Pi and BP represent the ith element (solution) and universal best element of the population, respectively, μGBP, μP, and σ are the constraints of the Gaussian Walk. Where μGBP is identical to BP and μP is identical to Pi, N is the total number of solutions in the group, $P_i'(j)$ is the new altered place of jth component of particle Pi, Pr and Pt are two randomly taken particles, $P_i^{"}$ is the new changed version of particle Pi, P_t' and $P_r'$are two randomly chosen particles [14] [17]. The standard deviation σ has been measured by equation (14).

$$\sigma = \left| \frac{\log(g)}{g} \times (P_i - BP) \right| \tag{14}$$

Where $\frac{\log(g)}{g}$ is used to execute a local hunt around every particle, the size of the Gaussian jump depends on this term. If its size decreases, then the number of generations g increases [15] [16].

Jaya Algorithm

The Jaya algorithm, originally conceived by Rao et al. [23], represents a global optimization strategy. It operates as an iterative learning approach based on a population, sharing fundamental characteristics with various other Evolutionary Computation (EC) methods. However, the distinctiveness of the Jaya algorithm lies in its pursuit of optimal results within a single phase, as opposed to employing genetic operations such as selection, crossover, and mutation on individuals. It's a straightforward concept and remarkable efficiency; the Jaya algorithm has developed as a highly appealing optimization technique. It has demonstrated successful application across a multitude of real-world challenges [18] [24].

Let f(x) be Kapur's fitness function to be maximized. At any cycle i, expect that there are ' m ' number of thresholds (for example, j=1,2,3,4,5), 'n' number of possible solutions (for example, population size, k=50). Let the best possible best get the best estimation of f(x) (for example, f(x)best) in the whole possible solutions and the worst estimation of f(x) (for example, f(x)worst) in the whole possible solutions [19] [25]. In the off chance that $X_{j,k,i}$ is the measurement of the jth variable for the kth solution

during the ith cycle, then this value is altered due to the accompanying (1). The scientific model of the Jaya algorithm is in equation (15):

$$X'_{j,k,i} = X_{j,k,i} + r_{1,j,i}\left(X_{j,best,i} - \left|X_{j,k,i}\right|\right) - \ r_{2,j,i}\left(X_{j,worst,i} - \left|X_{j,k,i}\right|\right) \qquad (15)$$

Where $X_{j,best,i}$ is the estimation of the variable j for the best solution, and $X_{j,worst,i}$ is the estimation of the variable j for the worst solution. $X_{j,k,}$ i is the updated estimation of $X_{j,k,}$ i, and $r_{1,j,i}$ and $r_{2,j,i}$ are the two random numbers for the jth variable during the ith cycle in the range [0, 1]. The expression $r_{1,j,i}\left(X_{j,best,i} - \left|X_{j,k,i}\right|\right)$ demonstrates the propensity of the solution for draw nearer to the best solution and the term $-r_{2,j,i}\left(X_{j,worst,i} - \left|X_{j,k,i}\right|\right)$ shows the inclination of the solution for keep away from the most exceedingly worst solution $X'_{j,k,i}$ is consented whether it gives better function value. All the consented function values toward the end of the cycle are kept up, and these qualities become the contribution to the following cycle [20] [21] [26]. The implementation steps of the Static Fractal Search (SFS) algorithm can be outlined as follows:

1. Initialization:

- Set algorithm parameters: Determine the highest number of repetitions, convergence criteria, and other control parameters.
- Initialize the population: Generate a set of solutions (individuals) within the problem's feasible region.
- Evaluate fitness: Calculate the fitness value for each solution using the main function of the problem.

2. Main Loop:

- For each iteration:
- Update the best values for all results based on their current positions.
- Identify the best and worst solutions within the existing population.

3. Improvement Phase:

- For each solution in the population:
- Adjust the solution's position towards the best solution, aiming for improvement.
- Ensure that the new position remains within the feasible region.

4. Exploration Phase:

- For every result in the population:
- Modify the solution's position away from the worst solution, promoting exploration.
- Keep the new position within the feasible region.

5. Update Best and Worst:

- After completing both improvement and exploration phases, identify the new best and worst solutions within the updated population.

6. Termination Criteria:

- Check convergence: Monitor whether the algorithm has converged based on predefined criteria, such as the highest number of repetitions or a small change in the finest fitness value.
- If convergence is reached or the maximum iterations are exceeded, proceed to the next step.

7. Solution Retrieval:

- Return the best solution (the solution with the lowest fitness value) as the optimized result to the problem.

8. Wrap-up:

- Analyse and interpret results: Evaluate the obtained solution, analyse convergence behaviour, and relate it to other optimization approaches if applicable.
- Fine-tuning: Adjust algorithm parameters if necessary to achieve better performance on similar problems.

EXPERIMENTS AND RESULTS IN SFS ALGORITHM

Within this section, outline the analytical criteria applied to assess the proposed method. Commencing with the presentation of benchmark images, subsequently provide a concise depiction of parameter configurations for both the SFS and Jaya algorithms. Subsequently, quality metrics are engaged to appraise the effectiveness of the thresholding process.

Benchmark Images

There are three commonly used benchmark test images that are widely recognized in the area of image processing: Cameraman, Lena, and Peppers. You can see each of these images separately in Figure 2. These benchmark images share certain characteristics:

Size: Each of these images is standardized to a size of 256x256 pixels, meaning they consist of 256 pixels in both the horizontal and vertical dimensions.

Gray Levels: They are all represented using 8-bit gray levels, which means there are 256 different shades of gray available in these images. These shades range from pure black (with a value of 0) to pure white (with a value of 255).

Experimental Settings

Within this section, experimentation is conducted on well-known grayscale benchmark images: Cameraman, Lena, and Peppers (as depicted in Figure 1).

Figure 2. The standard benchmark test images: cameraman, lena, and peppers

The evaluation of image thresholding performance is accomplished using the Jaccard metric [27]. Additionally, the application and effectiveness of the SFS algorithm in addressing multi-dimensional (MT) challenges are unveiled through its implementation on standard benchmark test images. The chosen parameter values for acquiring optimal threshold values are as follows: a population size of 50, a maximum diffusion count of 2, and an aggregate of 100 generations.

The Results and Discussions

The proposed multilevel thresholding framework was implemented in MATLAB (R2023a) in a high-performance workstation with an Intel Core i7 processor and 16GB of RAM. This background gave the solid computational libraries that were required to service 8-bit grayscale images and run the iterative cycles of the SFS and Jaya algorithms. Besides, the MATLAB Image Processing Toolbox supported statistical validation and calculation of image quality metrics, including Peak Signal-Noise Ratio (PSNR) and Structural Similarity Index (SSIM).

Peak Signal-to-Noise Ratio (PSNR)

The quality of reconstruction of the segmented picture is measured by PSNR. The larger the PSNR is, the better-quality image it represents, which is more similar to the original. It is calculated as in equation (16):

$$PSNR = 10\log_{10}\left(\frac{L^2}{MSE}\right)(dB) \tag{16}$$

Where L is the maximum intensity value (typically 255 for 8-bit images).

Structural Similarity Index (SSIM)

SSIM is used to measure the perceived quality of the image by comparing luminance, contrast, and structure. The value ranges from -1 to 1, where 1 indicates perfect structural similarity in equation (17).

$$SSIM_{(x,y)} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{17}$$

Where μ represents the average, σ^2 is the variance, and c are constants to stabilize the division.

SSIM is used to measure the perceived quality of the image by comparing luminance, contrast, and structure. The value ranges from -1 to 1, where 1 indicates perfect structural similarity in equation (18).

$$SSIM_{(x,y)} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{18}$$

Where μ represents the average, σ^2 is the variance, and c are constants to stabilize the division.

Given the stochastic nature of SFS, it becomes crucial to work on a suitable statistical metric to quantify its effectiveness. In order to maintain consistency with similar studies documented in the literature [7, 9], the experimentation involves varying the number of threshold levels, denoted by k, with values set at 2, 3, 4 & 5. The visualizations of Figure 1 are displayed at various threshold levels (k = 2, 3, 4 & 5) in Figure 3. These visualizations distinctly showcase the influence of the SFS algorithm on enhancing the superiority of the segmented image.

Founded on the data existing in Table 1, the SFS algorithm employs Otsu's function to compute optimal thresholds along with corresponding Jaccard measures across varying threshold levels (k = 2, 3, 4 & 5) for the three standard benchmark images. The outcomes are tabulated for analysis. Table 2 gives a summary of the evaluation process, showcasing the examination of the finest normal objective function

values across various threshold levels (k = 2, 3, 4 & 5). A higher average objective task value signifies superior thresholding performance. Notably, the values gained through SFS outperform those generated by alternative methods such as Darwinian Particle Swarm Optimization (DPSO), Hybrid Differential Evolution optimization (HDE), FA, IFA, Artificial Bee Colony (ABC), hybrid FA and ABC (FAABC), Social Spider Optimization (SSO), hybrid FA and SSO (FASSO), Sine Cosine Algorithm (SCA), hybrid ABC and SCA (ABCSCA), Hyper-heuristic Union Best (HHUB), and Hyper-heuristic Best (HHB).

The current approach focuses on computing variance by comparing pixel values with their corresponding average values. By maximizing variance, greater objective function values are achieved. Notably, a rise in the threshold level naturally leads to an increment in average objective function values. However, it's worth noting that a rise in threshold levels also leads to a greater number of function assessments. This explains the greater objective function values perceived in Table 2.

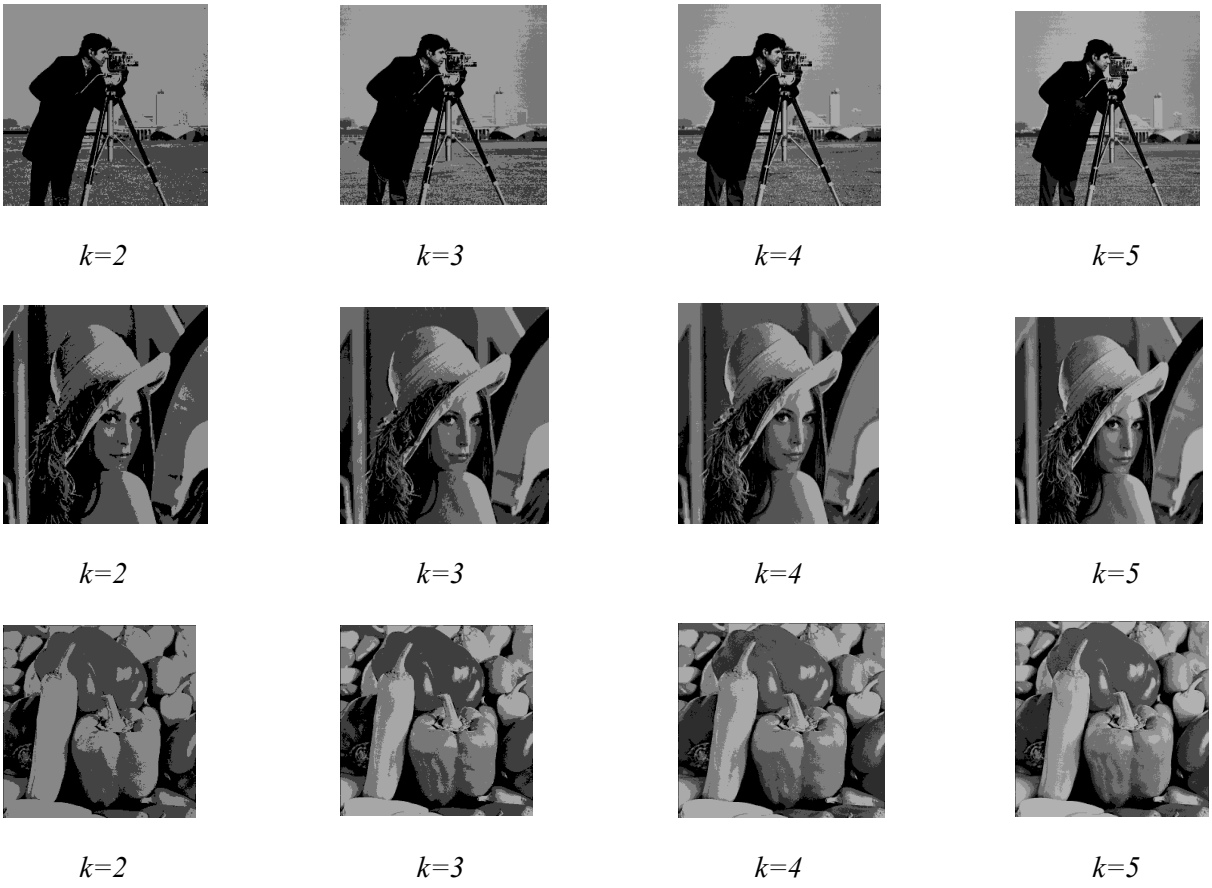

Figure 3. Segmented images with different thresholds levels k = 2, 3, 4 & 5 found by the SFS

Table 1. Optimum threshold and Jaccard measures gained by the SFS

Test Images	*k*	Thresholds	J_{ac}
Cameraman	2	70, 144	0.75
	3	59, 119, 156	0.76
	4	43, 95, 140, 170	0.78
	5	36, 82, 122, 149, 173	0.79
Lena	2	78, 145	0.63
	3	57, 106, 159	0.76
	4	47, 84, 119, 164	0.80
	5	46, 80, 109, 139, 174	0.80
Peppers	2	67, 135	0.83
	3	62, 119, 167	0.85
	4	45, 85, 126, 170	0.90
	5	43, 78, 112, 147, 177	0.90

Table 2. Comparison of objective function measures acquired utilizing various optimization algorithms

Test images	Various algorithms	Thresholds levels *k*			
		2	**3**	**4**	**5**
		Objective function measures			
Cameraman	DPSO [7]	3650.335	3725.715	3780.663	3811.995
	HDE [7]	3650.335	3725.715	3780.687	3811.969
	FA [7]	3650.335	3725.715	3780.687	3812.009
	IFA [7]	3650.335	3725.715	3780.687	3812.009
	ABC [9]	3572.100	3661.800	**4030.900**	**4069.700**
	FAABC [9]	3608.700	3681.000	3913.300	3985.100
	SSO [9]	3598.000	3682.000	3739.200	3773.100
	FASSO [9]	3613.200	3689.200	3729.700	3729.700
	SCA [9]	1410.700	0563.800	0536.900	0530.700
	ABCSCA [9]	3530.200	3572.600	3689.100	3847.700
	HHUB [9]	3651.900	3726.400	3780.500	3812.100
	HHB [9]	**3651.900**	**3727.400**	3782.300	3813.600
	SFS	3650.335	3725.715	3780.663	3812.009
Lena	DPSO [7]	1993.294	2162.980	2229.253	2253.910
	HDE [7]	1993.294	2162.980	2229.253	2253.905
	FA [7]	1993.294	2162.980	2229.253	2253.910
	IFA [7]	1993.294	2162.980	2229.253	2253.910
	ABC [9]	1873.600	2011.900	2100.000	2147.600
	FAABC [9]	1803.600	1900.800	1988.900	2001.800
	SSO [9]	1919.400	2040.600	2113.700	2146.500
	FASSO [9]	1928.100	2045.800	2109.700	2154.400
	SCA [9]	0110.800	0143.800	0153.600	0168.700
	ABCSCA [9]	1803.600	1900.800	1988.900	2001.800
	HHUB [9]	1964.400	2130.800	2191.500	2216.900
	HHB [9]	1964.400	2131.400	2194.800	2218.700
	SFS	**2341.164**	**2540.169**	**2617.225**	**2650.748**
Peppers	DPSO [7]	2532.321	2703.572	2766.459	2810.842
	HDE [7]	2532.321	2703.572	2766.459	2810.830
	FA [7]	2532.321	2703.572	2766.459	2810.842
	IFA [7]	2532.321	2703.572	2766.459	2810.842
	ABC [9]	2317.700	2466.500	2563.700	2637.800
	FAABC [9]	2368.900	2505.400	2574.400	2643.800
	SSO [9]	2391.700	2518.300	2582.900	2637.000
	FASSO [9]	2357.400	2510.800	2591.900	2628.100
	SCA [9]	1751.000	1863.300	1815.200	1800.400
	ABCSCA [9]	2333.400	2425.700	2476.400	2553.200
	HHUB [9]	2437.500	2588.200	2654.700	2694.700
	HHB [9]	2437.500	2588.300	2657.400	2696.700
	SFS	**2561.054**	**2734.706**	**2801.168**	**2846.798**

EXPERIMENTS AND RESULTS IN JAYA ALGORITHM

Benchmark Images

The set of six standard benchmark test images encompasses three frequently employed visuals: Cameraman, Peppers, Ostrich, Flower, Plane, and Girl, all depicted individually in Figure 4.

Figure 4. The standard benchmark test images: Cameraman, Peppers, Ostrich, Flower, Plane, and Girl

The dimensions of each benchmark image vary: Cameraman and Peppers are sized at 256x256 pixels, Ostrich and Flower at 321x481 pixels, and Plane and Girl at 481x321 pixels. All images exist in 8-bit Gray-level format.

Experimental Settings

In this section, a series of tests is conducted on benchmark grayscale images: Cameraman, Peppers, Ostrich, Flower, Plane, and Girl images. The evaluation of image thresholding performance is performed using the Jaccard metric [28]. The application and effectiveness of the Jaya algorithm in addressing multi-dimensional (MT) challenges are demonstrated through its implementation on standard benchmark test images [29] [30]. The parameters selected for obtaining optimal threshold values include a population size of 50 and a maximum number of generations set to 100 [31] [32].

The Results and Discussions

The randomness in the operations of the Jaya algorithm makes it essential that there is a careful evaluation of its performance in any application that is to be put into use. To ensure that they adhere to the known methodologies in research, as has been described in the literature [9], the experimentation plan is based on the manipulation of one of the key parameters, which is referred to as k. In particular, examine how the change in the number of threshold levels, which is shown by this parameter, can be examined with the values between 1 and 5.

Through controlled variation of the parameter of granularity or complexity (k), the experiment explores how various degrees of complexity or granularity can affect the performance of the Jaya algorithm when applied in image segmentation. In this way, are able to discover the manner in which the algorithm changes and adjusts to different levels of complexity in the task. By so doing, are in a position to understand very well its malleability and strength under a variety of conditions that can in turn lead to a more detailed understanding of its strength and its weaknesses. Moreover, this form of experiment will boost the validity of research results, as well as making it easy to compare with other research projects conducted in the past that adhered to similar designs. This consistency and reproducibility guarantee that the evaluation of the performance of the Jaya algorithm will be well-based on well-established practices. Finally, the study offers a sound background in the knowledge of the potential and constraints of the algorithm, which will serve as a good source of information on the practical usage as well as the future advancement.

Figure 4 presents a collection of refined visualizations that were developed based on exploration of the data and findings at the specific threshold, namely, k = 1, 2, 3, 4, and 5. Such visualizations can be viewed as an effective instrument in explaining how the implementations of the Jaya algorithm have a

considerable effect on the nature and the quality of the segmented images. They offer a very informative and graphically intuitive way of seeing how the algorithm affects image segmentation, and as such, can value the performance variations of the algorithm as a range of threshold values are varied.

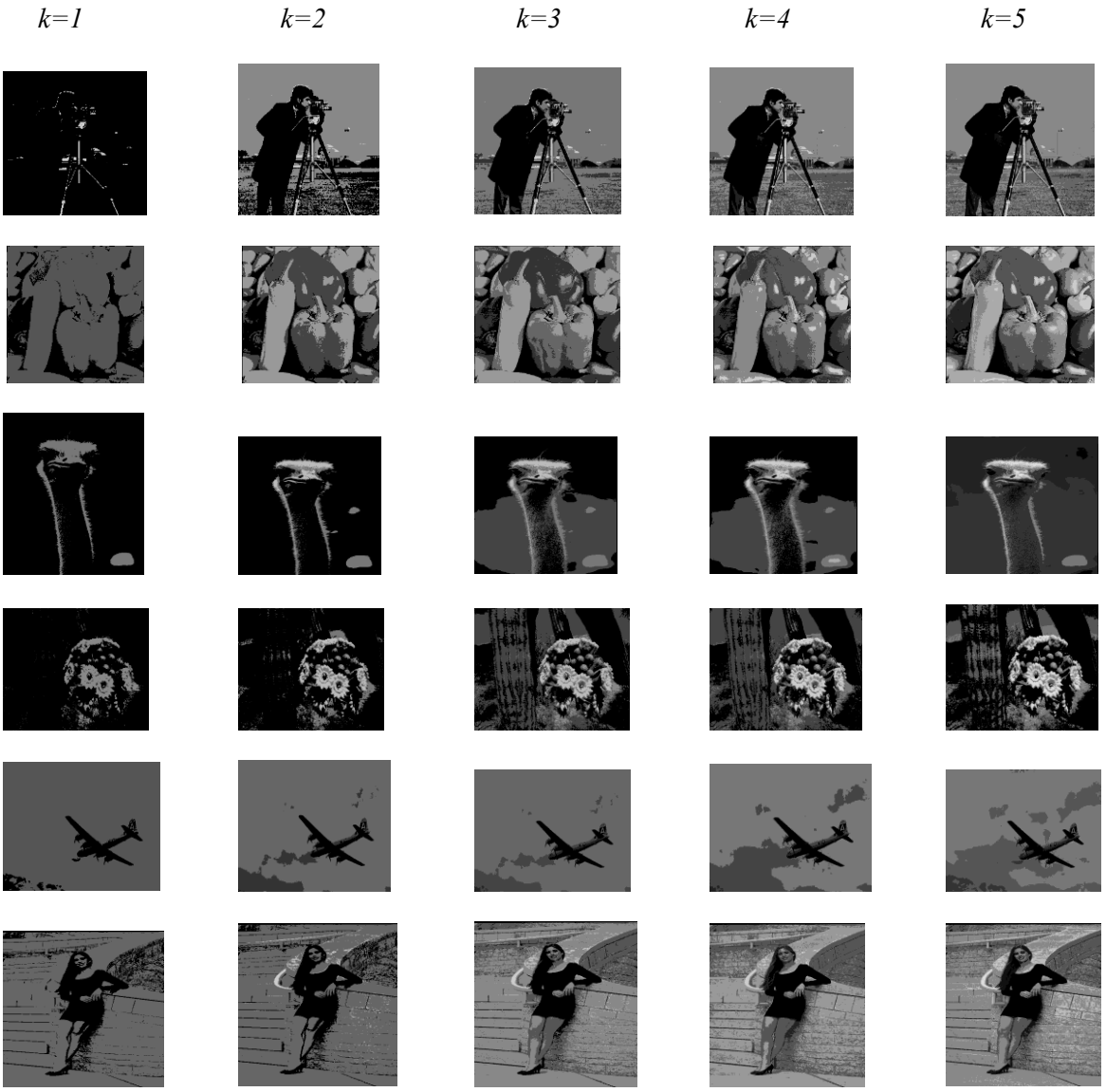

Figure 5. Segmented images with threshold levels k = 1, 2, 3, 4 & 5 obtained by the Jaya algorithm

Table 3. Optimal threshold and Jaccard measures gained by the jaya algorithm

Test images	k	Thresholds	Jac	Test images	k	Thresholds	Jac
Cameraman	1	193	0.02	Flower	1	137	0.10
	2	128, 193	0.60		2	118, 181	0.16
	3	40, 102, 196	0.78		3	79, 130, 186	0.50
	4	48, 97, 150, 198	0.78		4	71, 118, 158, 209	0.58
	5	31, 65, 99, 149, 197	0.79		5	67, 119, 149, 182, 215	0.61
Peppers	1	80	0.78	Plane	1	84	0.93
	2	74, 147	0.81		2	66, 101	0.95
	3	56, 109, 163	0.87		3	34, 71, 103	0.96
	4	52, 103, 150, 192	0.88		4	34, 77, 101, 159	0.96
	5	34, 75, 110, 152, 191	0.92		5	39, 77, 104, 129, 161	0.96
Ostrich	1	127	0.08	Girl	1	109	0.78
	2	119, 180	0.10		2	106, 202	0.80
	3	78, 122, 183	0.51		3	95, 144, 202	0.85
	4	29, 81, 123, 189	0.99		4	41, 85, 141, 201	0.91
	5	31, 83, 126, 167, 203	0.99		5	47, 89, 134, 180, 209	0.91

The significance of the inclusion of the Jaccard measures is paramount since the measure provides a quantitative value of the level of overlap and agreement between the segmented results produced by the algorithm and the ground truth data. The Jaccard measures, by quantifying the accuracy and similarity of the results that are segmented, give evaluation strong numerical basis so that can make well-founded judgement regarding the performance of the algorithm. The objective function measures are an important characteristic, which is the capacity of the algorithm to obtain the best segmentation results. This table summarizes the main findings and gives us an opportunity to draw conclusions about the threshold level or levels that will give the most desirable results of segmentation. It is, in fact, a summed-up description of the efficiency of the algorithm at different thresholding conditions.

Table 4. Comparison of objective function values acquired utilizing various optimization algorithms

Test Images	**Various algorithms**	**Thresholds levels *k***				
		1	**2**	**3**	**4**	**5**
		Objective function measures				
Cameraman	PSO [9]	8.7868	12.2865	15.3744	18.5567	21.2809
	FA [9]	8.7748	12.2865	15.3928	18.5563	21.3213
	ABC [9]	8.7868	12.2865	15.3927	18.5445	21.2756
	GA [9]	8.7747	12.2865	15.381	18.5564	21.2792
	GWO [9]	8.7868	12.2865	15.3942	18.5545	21.3027
	SOS [9]	**8.7868**	**12.2865**	**15.3943**	**18.5567**	21.3254
	Jaya	8.7179	12.1688	15.2145	18.4524	**21.3320**
Peppers	PSO [9]	9.1423	12.6346	15.6887	18.5216	21.2730
	FA [9]	9.1423	12.6346	15.6887	18.5354	21.2817
	ABC [9]	9.1423	12.6346	15.6885	18.5238	21.2446
	GA [9]	9.1423	12.6346	15.6883	18.5229	21.2755
	GWO [9]	9.1423	12.6346	15.6886	18.5354	21.2766
	SOS [9]	9.1423	12.6346	15.6887	18.5392	21.2818
	Jaya	**9.1700**	**12.6782**	**15.7544**	**18.6206**	**21.3413**
Ostrich	PSO [9]	9.0648	12.5935	15.655	18.5555	21.3769
	FA [9]	9.0648	12.5935	15.655	18.5555	21.4604
	ABC [9]	9.0648	12.5935	15.654	18.5476	21.3940
	GA [9]	9.0648	12.5935	15.6547	18.5528	21.4068
	GWO [9]	9.0648	12.5935	15.6548	18.547	21.4547
	SOS [9]	9.0648	12.5935	15.6550	18.5563	21.4613
	Jaya	**9.0728**	**12.6125**	**15.6671**	**18.5708**	**21.4938**
Flower	PSO [9]	9.2252	12.6227	15.7331	18.6951	21.3700
	FA [9]	9.2252	12.6227	15.7369	18.6949	21.3716
	ABC [9]	9.2252	12.6227	15.7364	18.6896	21.3488
	GA [9]	9.2252	12.6227	15.7364	18.6936	21.3670
	GWO [9]	9.2252	12.6227	15.7362	18.6941	21.3677
	SOS [9]	9.2252	12.6227	15.7369	18.6951	21.3719
	Jaya	**9.2911**	**12.7610**	**15.9050**	**18.8861**	**21.5326**
Plane	PSO [9]	8.1580	11.0739	13.8912	16.6455	19.1482
	FA [9]	8.1580	11.0774	13.9522	16.6648	19.1448
	ABC [9]	8.1580	11.0774	13.9571	16.6311	19.0740
	GA [9]	8.1580	11.0758	13.9406	16.639	19.1279
	GWO [9]	8.1580	11.0774	13.9574	16.6497	19.1290
	SOS [9]	8.1580	11.0774	13.9586	16.6705	19.1478
	Jaya	**8.2231**	**11.1549**	**14.0286**	**16.7154**	**19.1490**
Girl	PSO [9]	8.6091	11.9353	15.0761	17.874	20.6819
	FA [9]	8.6091	11.934	15.0761	17.8733	20.6940
	ABC [9]	8.6091	11.9353	15.0751	17.8607	20.6371
	GA [9]	8.6091	11.9353	15.076	17.8727	20.6716
	GWO [9]	8.6091	11.9353	15.0735	17.8695	20.6873
	SOS [9]	8.6091	11.9353	15.0761	17.8735	20.6977
	Jaya	**8.6681**	**12.0151**	**15.1831**	**18.0334**	**20.8251**

Together, Figure 5, Table 3, and Table 4 incorporated into the study provide a complete picture of the results. It enables us not only to view the results of the segmentation in a visual way but also to evaluate the performance of an algorithm in terms of quantitative measures of various threshold values. It is fascinating to note that the values revealed by the Jaya algorithm are more valuable in comparison with the values obtained by other methods like Particle Swarm Optimization (PSO), Firefly Algorithm (FA), Artificial Bee Colony (ABC), Genetic Algorithm (GA), the use of the grey wolf as an optimizer (GWO), and the use of symbiotic organisms as a search (SOS). These are all the algorithms that are embraced to make a fair comparison, as shown in [9]. The aim of this approach is to augment the entropy and statistical dispersion of the image histogram. Optimizing entropy gives higher values of objective functions. It is noteworthy that there is a natural increase in average objective function values with an increase in threshold level. Nevertheless, it should be noted that the increase in threshold levels also leads to an increase in the number of function assessments. This is due to the fact that the average values of objective functions are greater in Table 4.

The enhanced visual representations of Figure 3, presented at distinct threshold levels k = 1, 2, 3, 4 & 5, are showcased in Figure 4. These visualizations effectively illustrate how the application of the Jaya algorithm contributes to the nature of the segmented image. From the data tabulated in Table 3, the Jaya algorithm, utilizing the Kapur function, computes optimal thresholds along with corresponding Jaccard measures across various threshold levels k = 1, 2, 3, 4 & 5 for the standard benchmark test images. Moreover, Table 4 outlines the evaluation of the finest average objective function values at the aforementioned threshold levels. A higher average objective function measure signifies superior thresholding performance. The quality assessment of the algorithm's multi-threshold selection is based on the utilization of the Jaccard measure.

Ablation Study, to evaluate the individual contributions of the algorithm components, an ablation study was performed by isolating the Diffusion Process in SFS and the Attraction-Repulsion logic in Jaya. Findings revealed that disabling the SFS diffusion process led to a decrease in PSNR by approximately 5–8% due to premature convergence. Similarly, removing the worst-solution avoidance term from the Jaya algorithm increased the standard deviation of fitness values, indicating reduced stability. These results confirm that the specific exploration mechanisms of SFS and the parameter-free movement of Jaya are essential for achieving optimal multilevel thresholds compared to standard versions of the algorithms.

CONCLUSION

This study has been able to establish the effectiveness of the Stochastic Fractal Search (SFS) algorithm and the Jaya algorithm in multilevel image thresholding (MT) challenges. These metaheuristic methods, combined with Otsu between-class variance and Kapur Entropy criteria, are used in the research to create a powerful model of image segmentation improvement. The experimental findings on standard 256 × 256 8-bit benchmark images (Cameraman, Lena, and Peppers) indicate that both algorithms have a tendency to reach global optima. Statistically speaking, the methods that are proposed have high values of objective functions even as the number of threshold levels (K) goes higher, beyond 2 to 5. The Jaya and the SFS algorithm have a linear-like computational growth as opposed to the traditional exhaustive search methods that exhibit exponential-like growth. Comparative analysis indicates that the Jaya algorithm, which is parameter-free, is able to achieve these results with much lower processing time and much greater stability, as indicated by a lower standard deviation with different trial runs, as compared to Genetic Algorithms (GA) and Particle Swarm Optimization (PSO). The accuracy of the maximization of between-class variance and entropy makes the segments coming out of it discrete and representative of the original data. This renders the method very appropriate in clinical settings with high stakes, like medical diagnostic imaging, where classification of clinical images into positive and negative diagnoses is very crucial in informed decision-making. The future directions will be to consider the incorporation of more types and forms of entropy, e.g., Tsallis or Rényi entropy, to achieve a greater adaptability in segmentation to different textures of an image. Further, it is possible to apply these algorithms to the 3D image sphere and real-time video processing, which will promote the development of industries with the use of autonomous navigation for sophisticated satellite remote sensing.

REFERENCES

[1] Dey S, Bhattacharyya S, Maulik U. Quantum behaved multi-objective PSO and ACO optimization for multi-level thresholding. In2014 International Conference on Computational Intelligence and Communication Networks 2014 Nov 14 (pp. 242-246). IEEE. https://doi.org/10.1109/CICN.2014.63

[2] Bhandari AK, Kumar A, Singh GK. Tsallis entropy based multilevel thresholding for colored satellite image segmentation using evolutionary algorithms. Expert systems with applications. 2015 Dec 1;42(22):8707-30. https://doi.org/10.1016/j.eswa.2015.07.025

[3] Otsu N. A threshold selection method from gray-level histograms. Automatica. 1975 Jun;11(285-296):23-7.

[4] Kapur JN, Sahoo PK, Wong AK. A new method for gray-level picture thresholding using the entropy of the histogram. Computer vision, graphics, and image processing. 1985 Mar 1;29(3):273-85. https://doi.org/10.1016/0734-189X(85)90125-2

[5] Li K, Tan Z. An improved flower pollination optimizer algorithm for multilevel image thresholding. IEEE access. 2019 Nov 14;7:165571-82. https://doi.org/10.1109/ACCESS.2019.2953494

[6] Zarezadeh S, Asadi M. Results on residual Rényi entropy of order statistics and record values. Information Sciences. 2010 Nov 1;180(21):4195-206. https://doi.org/10.1016/j.ins.2010.06.019

[7] Chen K, Zhou Y, Zhang Z, Dai M, Chao Y, Shi J. Multilevel image segmentation based on an improved firefly algorithm. Mathematical Problems in Engineering. 2016;2016(1):1578056. https://doi.org/10.1155/2016/1578056

[8] Abd El Aziz M, Ewees AA, Hassanien AE. Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. Expert Systems with Applications. 2017 Oct 15;83:242-56. https://doi.org/10.1016/j.eswa.2017.04.023

[9] Abd Elaziz M, Ewees AA, Oliva D. Hyper-heuristic method for multilevel thresholding image segmentation. Expert Systems with Applications. 2020 May 15;146:113201. https://doi.org/10.1016/j.eswa.2020.113201

[10] Sharma A, Chaturvedi R, Kumar S, Dwivedi UK. Multi-level image thresholding based on Kapur and Tsallis entropy using firefly algorithm. Journal of Interdisciplinary Mathematics. 2020 Feb 17;23(2):563-71. https://doi.org/10.1080/09720502.2020.1731976

[11] Küçükuğurlu B, Gedikli E. Symbiotic organisms search algorithm for multilevel thresholding of images. Expert Systems with Applications. 2020 Jun 1;147:113210. https://doi.org/10.1016/j.eswa.2020.113210

[12] Salimi H. Stochastic fractal search: a powerful metaheuristic algorithm. Knowledge-based systems. 2015 Feb 1;75:1-8. https://doi.org/10.1016/j.knosys.2014.07.025

[13] Hinojosa S, Dhal KG, Abd Elaziz M, Oliva D, Cuevas E. Entropy-based imagery segmentation for breast histology using the stochastic fractal search. Neurocomputing. 2018 Dec 10;321:201-15. https://doi.org/10.1016/j.neucom.2018.09.034

[14] Beken K, Caddwine H. Data-Efficient Learning-Assisted Predictive Control for Real-Time Trajectory Planning Under Dynamic Constraints. Journal of Scalable Data Engineering and Intelligent Computing. 2026 Jan 10:17-23.

[15] Dhal KG, Gálvez J, Ray S, Das A, Das S. Acute lymphoblastic leukemia image segmentation driven by stochastic fractal search. Multimedia Tools and Applications. 2020 May;79(17):12227-55. https://doi.org/10.1007/s11042-019-08417-z

[16] Bingöl O, Paçacı S, Güvenç U. Entropy-based skin lesion segmentation using stochastic fractal search algorithm. InThe International Conference on Artificial Intelligence and Applied Mathematics in Engineering 2019 Apr 20 (pp. 801-811). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-36178-5_69

[17] Shan S, Zhao F, Li Z, Luo L, Li X. A Comprehensive Review of Optical Metrology and Perception Technologies. Sensors. 2025 Nov 7;25(22):6811. https://doi.org/10.3390/s25226811

[18] Khalilpourazari S, Khalilpourazary S. A Robust Stochastic Fractal Search approach for optimization of the surface grinding process. Swarm and Evolutionary Computation. 2018 Feb 1; 38:173-86. https://doi.org/10.1016/j.swevo.2017.07.008

[19] Mellal MA, Zio E. A penalty guided stochastic fractal search approach for system reliability optimization. Reliability Engineering & System Safety. 2016 Aug 1; 152:213-27. https://doi.org/10.1016/j.ress.2016.03.019

[20] Chen X, Yue H, Yu K. Perturbed stochastic fractal search for solar PV parameter estimation. Energy. 2019 Dec 15;189:116247. https://doi.org/10.1016/j.energy.2019.116247

[21] Alomoush MI, Oweis ZB. Environmental-economic dispatch using stochastic fractal search algorithm. International Transactions on Electrical Energy Systems. 2018 May;28(5):e2530. https://doi.org/10.1002/etep.2530

[22] El-Kenawy ES, Eid MM, Saber M, Ibrahim A. MbGWO-SFS: Modified binary grey wolf optimizer based on stochastic fractal search for feature selection. IEEE Access. 2020 Jun 9; 8:107635-49. https://doi.org/10.1109/ACCESS.2020.3001151

[23] Khalilpourazari S, Naderi B, Khalilpourazary S. Multi-objective stochastic fractal search: A powerful algorithm for solving complex multi-objective optimization problems. Soft Computing. 2020 Feb;24(4):3037-66. https://doi.org/10.1007/s00500-019-04080-6

[24] Lin J, Wang ZJ. Multi-area economic dispatch using an improved stochastic fractal search algorithm. Energy. 2019 Jan 1; 166:47-58. https://doi.org/10.1016/j.energy.2018.10.065

[25] Al-Yateem N, Ismail L, Ahmad M. A comprehensive analysis on semiconductor devices and circuits. Progress in Electronics and Communication Engineering, 2 (1), 1–15 [Internet]. 2024

[26] Rao R. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations. 2016 Jan 1;7(1):19-34. https://doi.org/10.5267/j.ijiec.2015.8.004

[27] Rao RV, More KC. Design optimization and analysis of selected thermal devices using self-adaptive Jaya algorithm. Energy Conversion and Management. 2017 May 15; 140:24-35. https://doi.org/10.1016/j.enconman.2017.02.068

[28] Rahim R. Scalable architectures for real-time data processing in IoT-enabled wireless sensor networks. Journal of Wireless Sensor Networks and IoT. 2024;1(1):28-31. https://doi.org/10.31838/WSNIOT/01.01.07

[29] Wang S, Rao RV, Chen P, Zhang Y, Liu A, Wei L. Abnormal breast detection in mammogram images by feed-forward neural network trained by Jaya algorithm. Fundamenta Informaticae. 2017 Mar 11;151(1-4):191-211. https://doi.org/10.3233/FI-2017-1487

[30] Marangunic C, Cid F, Rivera A, Uribe J. Machine Learning Dependent Arithmetic Module Realization for High-Speed Computing. Journal of VLSI circuits and systems. 2022 Sep 28;4(01):42-51. https://doi.org/10.31838/jvcs/04.01.07s

[31] Rao RV, Rai DP, Balic J. Surface grinding process optimization using Jaya algorithm. In Computational Intelligence in Data Mining—Volume 2: Proceedings of the International Conference on CIDM, 5-6 December 2015 2015 Dec 10 (pp. 487-495). New Delhi: Springer India. https://doi.org/10.1007/978-81-322-2731-1_46

[32] El Aziz MA, Ewees AA, Hassanien AE. Hybrid swarms optimization based image segmentation. In Hybrid soft computing for image segmentation 2016 Nov 13 (pp. 1-21). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-47223-2_1