A study of Chaotic maps in Differential Evolution applied to gray-level Image Thresholding

Uroš Mlakar, Janez Brest, Iztok Fister Jr., Iztok Fister
Faculty of Electrical Engineering and Computer Science
University of Maribor, Slovenia
Email: uros.mlakar@um.si

***Abstract*—Image segmentation is an important preprocessing step in many computer vision applications, using the image thresholding as one of the simplest and the most applied methods. Since the optimal thresholds' selection can be regarded as an optimization problem, it can be found easily by applying any meta-heuristic with an appropriate objective function. This paper investigates the impact of different chaotic maps, embedded into a self-adaptive differential evolution for the purpose of image thresholding. The Kapur entropy is used as an objective function that maximizes the entropy of different regions in the image. Three chaotic maps, namely the Kent, Logistic and Tent, found commonly in literature, are studied in this paper. The applied chaotic maps are compared to the original differential evolution, self-adaptive differential evolution, and the state-of-the-art L-Shade tested on four images. The results show that the applied chaotic maps improve the results obtained using the traditional randomized method.**

I. Introduction

Image segmentation is a process of dividing an image into disjoint sets, which share similar properties, such as intensity or color. Image segmentation is normally the first step in many applications of computer vision, such as feature extraction, image recognition, and classification of objects. Simply put, it is a process of dividing an image into regions, which are then processed further by higher level methods. Image thresholding is one of the most simple segmentation methods performing the image segmentation based on values contained in the image histogram. The histogram is a probability distribution of the colors contained in the image that must be calculated prior to the segmentation process. Image thresholding is one of the most used methods for image preprocessing, because of its simplicity. In the case of separating an image into two disjoint regions, the process is called bi-level thresholding, while we deal with multilevel thresholding, when separating the image into several regions. The selection of optimal threshold values is crucial, since the results of a good segmentation are a prerequisite for further image processing.

Recently, we can see a trend of growing interest for using the thresholding methods. Thus, the optimal threshold selection can be formulated as an optimization problem. In line with this, different optimization criteria are used, like class variance or various entropy measures. As a result, many meta-heuristics have been developed for solving this problem, since the exhaustive methods become computationally expensive, especially, when confronted with the larger number of thresholds. On the other hand, there is also a rising interest in studying chaotic systems applied to various meta-heuristic algorithms. It has been proven that, when enhancing a meta-heuristic algorithm with a chaotic map, its convergence property is improved, since it offers a better exploitation of the current solutions [8].

In this paper, we want to investigate the influence of different chaotic maps, when applied to a self-adaptive differential evolution algorithm (jDE) [3] for solving the multi-level gray-scale image thresholding. The chaotic enhanced algorithms are compared with the standard Differential Evolution (DE) [18] and the state-of-the-art L-Shade [20] tested on four standard test images found in literature.

The remainder of this paper is structured as follows. In Section II, a brief review of novel meta-heuristics is presented applied to image thresholding is presented. Then, Section III describes the Differential Evolution (DE) algorithm and its extension (jDE). In Section IV, the studied chaotic maps are presented, while Section V defines the image thresholding problem formally. The results of experimentation are gathered in Section VI. The paper outlines the future directions of its development, in Section VII.

II. Related Work

Recently, there is growing interest in using various meta-heuristics for image segmentation. Actually, the exhaustive methods were demonstrated to be computationally ineffective in practice. Therefore, the researchers search for new ways for solving this problem.

Sarkar et al. [17] utilized a minimum cross entropy method for objective function in solving the image segmentation with a DE algorithm. Thus, a comparison with several meta-heuristics and an exhaustive search was performed, where the DE clearly outperformed the other methods used in the study. In another study by Sarkar et al. [16], the DE using the Tsallis entropy as an objective function was studied, giving the best results among the comparing algorithms. Cuevas et al. [4] used the DE for finding a mix of Gaussian functions, which approximated the given histogram of the input image as closely as possible. Their results stated that the method is feasible for fast and reliable image segmentation.

Suresh and Lal [19] presented a computationally efficient Cuckoo Search (CS) algorithm for satellite image segmentation. They compare their CS variant to other meta-heuristic methods based on Otsu's between-class variance, and Kapur's and Tsallis' entropies. Their proposed algorithm outperformed the others in attaining the global optimum thresholds as well as the convergence rate. A CS for multilevel satellite image segmentation is presented in [2], where the proposed algorithm also provides good results by selecting the optimal thresholds effectively and properly.

Alidhozic and Tuba [1] improved the bat algorithm with some elements of the DE and Artificial Bee Colony (ABC) algorithms. They compare their improved bat algorithm with other state-of-the-art algorithms, where the results showed a significant improvement in the convergence speed, and also improving the quality of the results. A maximum entropy thresholding method, aided with the ABC algorithm was studied by Horng [11]. The results show that the method achieves similar results as the PSO, hybrid PSO, fast Otsu's method, and honey bee mating optimization algorithm. Again, the computational time is reduced when using the proposed algorithm.

III. Differential evolution

Differential evolution (DE) [18] is a population based algorithm designed for global optimization. It belongs to the family of Evolutionary Algorithms (EAs), since it mimics a complex evolutionary process using simple mathematical equations. The original DE maintains a population of NP solutions $\mathbf{x}_i = \{x_{ij}\}$, for $i = 1, \ldots, NP \ \wedge j = 1, \ldots, D$, which are improved using various evolutionary operators during each generation g. By using the operators, like mutation, crossover, and selection, a trial vector (offspring) is produced, which competes with its parent for survival. Thus, the better between trial and parent solution, accordingly the fitness value is selected to undergo the evolutionary process in the succeeding generation $g + 1$.

In DE, a mutant vector is created by applying a mutation strategy for each population vector $\mathbf{x}_i$. There are many different mutation strategies found in the literature, but for the purpose of this study the 'best/1/bin' strategy was applied as follows:

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{best} + F(\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}), \tag{1}$$

where the r_1 and r_2 are random integers from the interval $1, \ldots, NP$ and the following inequality holds $r_1 \neq r_2 \neq i$. Factor F is used to control the amplification of the difference vector, and is defined mostly within the interval $[0, 1]$. The next step in the evolutionary loop is the recombination of the newly created mutant vector $\mathbf{v}_{i,g+1}$ with the target vector $\mathbf{x}_{i,g}$ to create a trial vector by using a crossover:

$$u_{ij,g+1} = \begin{cases} v_{ij,g+1}, & \text{if } rand(0,1) \leq Cr \text{ or } j = j_{rand}, \\ x_{ij,g+1}, & \text{otherwise.} \end{cases} \tag{2}$$

As can be seen from Eq. (2), the crossover rate Cr is defined at the interval $[0, 1]$ and it defines the probability of modifying the corresponding element of the trial vector with $u_{ij,g}$. The j_{rand} index is responsible for the trial vector to contain at least one value from the mutant vector; this mechanism is employed to prevent the cloning of target vectors. In the original DE the control parameters are fixed as $F = 0.5$, and $Cr = 0.9$, during the evolutionary process.

An extension of the DE algorithm was proposed in [3], which self-adapts the F and Cr control parameters. When generating the $\mathbf{v}_{i,g+1}$ and $\mathbf{u}_{i,g+1}$ vectors, firstly the corresponding F_i and Cr_i are updated using the following mechanisms:

$$F_{i,g+1} = \begin{cases} F_l + rand_1 F_u, & \text{if } rand_2 \leq \tau_1, \\ F_{i,g}, & \text{otherwise.} \end{cases} \tag{3}$$

$$Cr_{i,g+1} = \begin{cases} rand_3, & \text{if } rand_4 \leq \tau_2, \\ Cr_{i,g}, & \text{otherwise.} \end{cases} \tag{4}$$

The $rand_j$ for $j = 1, \ldots, 4$ are randomly generated numbers from the interval $[0, 1]$ and $\tau_1 = \tau_2 = 0.12$ [22].

Finally, the selection operator compares the fitness function value of the trial vector $\mathbf{u}_{i,g+1}$ with the same value of the target vector $\mathbf{x}_{i,g}$. The fittest vector is selected to undergo the evolutionary process in the next generation:

$$F_{i,g+1} = \begin{cases} \mathbf{u}_{i,g+1} & \text{if } f(\mathbf{u}_{i,g+1}) \geq f(\mathbf{x}_{i,g}), \\ \mathbf{x}_{i,g}, & \text{otherwise.} \end{cases} \tag{5}$$

IV. Chaotic maps

Recently, many new applications of EAs combined with chaotic maps have been reported in literature [15], [22]. The results of the experiments revealed that applying the chaotic maps in an EA increases the exploitation of the current solutions which, in general, improves the convergence property of the algorithm. Thus, the concepts of chaos applied to an EA would be beneficial. Although many researchers focus on applying the chaos for updating the parameters during algorithm runs, we investigated the behavior of the algorithms, when the random number generator is completely replaced with a chaotic map.

A lot of chaotic maps have been introduced in the literature, applicable to different domains of human activity. In this paper we investigate three chaotic maps, which are described more thoroughly in the following subsections.

A. Kent map

Kent map [7] is among the most studied chaotic maps, used in many applications, such as encryption, and can be expressed as:

$$x_{n+1} = \begin{cases} \frac{x_n}{m}, & 0 < x_n \leq m, \\ \frac{1-x_n}{1-m}, & m < x_n < 1, \end{cases} \tag{6}$$

where $0 < m < 1$. If $x_0 \in [0, 1]$, for all $n \geq 1$, $x_n \in [0, 1]$.

B. Logistic map

The logistic map [7] is defined by the following iterated function:

$$x_{n+1} = rx_n(1 - x_n), \tag{7}$$

where $x_n \in [0, 1]$ and r is a parameter. The generated time series are chaotic, when the iterated Logistic map with $r = 4$ is used.

C. Tent map

Tent map [14] is generated according to the following iterated function:

$$x_{n+1} = \begin{cases} \mu x_n, & x_n < \frac{1}{2}, \\ \mu(1 - x_n), & x_n \geq \frac{1}{2}, \end{cases} \tag{8}$$

where for $\mu = 2$, the tent map is a non-linear transformation of both the bit shift map and the $r = 4$ case of the logistic map [7].

V. Image thresholding

Image thresholding is the most simple and common method for image segmentation. In gray-scale images, the thresholds define the intensity values for classifying the image into different groups. The thresholding is divided into bi-level and/or multi-level, based on the number of prescribed thresholds.

Consider a gray-scale image with intensity values ranging from 0 to $N-1$, where N is the maximum possible intensity value. When considering bi-level thresholding, the goal is to find the intensity value, which makes the foreground and background regions the most distinguishable. Several methods have been adopted for this task, where many of them rely on calculating the variances of the pixel values in distinct regions, or calculating various entropy measures for the objective function.

When an image contains multiple regions, which cannot be separated by a single threshold, bi-level thresholding fails to provide a good solution. Hence, we must apply a multilevel thresholding method, which in most cases is just a simple extension of the bi-level thresholding scheme.

A. Kapur's entropy

Image thresholding on Kapur's entropy bases on the fact that an image is comprised of a background and foreground regions, which contribute to the probability distribution of the intensity values in the image [12]. The entropy of each region is calculated independently, while the sum of entropy values needs to be maximized. This maximization can be regarded as an optimization problem formulated as follows:

$$[T_1, \ldots, T_t] = \arg \ \max \sum_{i=1}^{t} H_i, \tag{9}$$

$$H_i = - \sum_{j=t_i}^{t_{i+1}-1} (\frac{p_j}{\omega_i}); \quad \omega_i = \sum_{j=t_i}^{t_{i+1}-1} p_j \tag{10}$$

Here the H_i denotes the entropy value for the i-th threshold, and p_j is the probability of the pixel intensity value.

In this paper the Kapur's entropy is utilized as the objective function for searching the optimal thresholding.

VI. Results

A. Image dataset

To conduct the experiments, we have chosen four standard images usually found in the literature from the computer vision field [21]. All images are of the size 256×256 pixels. The images and their corresponding histograms are depicted in Fig. 1. It is evident that the histograms of the images are multi-modal, which makes the task of the optimal thresholds' selection additionally difficult.

B. Experimental settings

In order to study the impact of chaotic maps on gray-scale image thresholding, the following algorithms were taken into consideration: original DE, jDE, jDE with Kent chaotic map (jDE$_{Kent}$), jDE with Logistic chaotic map (jDE$_{Log}$), jDE with Tent map (jDE$_{Tent}$) and, lastly, the state-of-the-art L-Shade [20]. The implementation for DE and jDE algorithms were provided on our own, while the code for L-Shade was taken from the CEC 2014 competition website. To provide the comparison of the observed algorithms as fairly as possible, the stopping criteria for all algorithms' runs was set to 10,000 function evaluations with a total of 30 runs per algorithm. The population size was fixed at 20 for all algorithms, except for L-Shade, whose parameters were kept as provided in the implementation. All experiments were performed on the test images using the 2, 4, 6, 8, 10, and 12 threshold levels.

C. Performance metrics

For evaluating the quality of segmentation results, two established performance metrics such as $PSNR$ and $SSIM$ were considered to compare the results of the algorithms in the study. Additionally, the required CPU computational time of algorithms was used for searching the optimal thresholds. The accuracy of the reconstructed image is measured using the measure $PSNR$ of the segmented images, since this measure relies directly upon the pixel intensity values. The $PSNR$ can be expressed mathematically as:

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right), \tag{11}$$

where MSE is defined as:

$$MSE = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \left[I(i,j)) - J(i,j) \right]^2. \tag{12}$$

Variables I and J in Eq. (12) are the original and segmented images, respectively.

On the other hand, $SSIM$ provides an assessment on image quality based on the degradation of structural information that is calculated as:

$$SSIM(I,J) = \frac{(2\mu_I \mu_J + C_1)(2\sigma_{IJ} + C_2)}{(\mu_I^2 + \mu_J^2 + C_1)(\sigma_I^2 + \sigma_J^2 + C_2)}, \tag{13}$$

where μ_x and μ_y stand for the mean intensities of images I and J, σ_x and σ_y represent standard deviations of I and J, and σ_{xy} is the local correlation coefficient between I and J. C_1

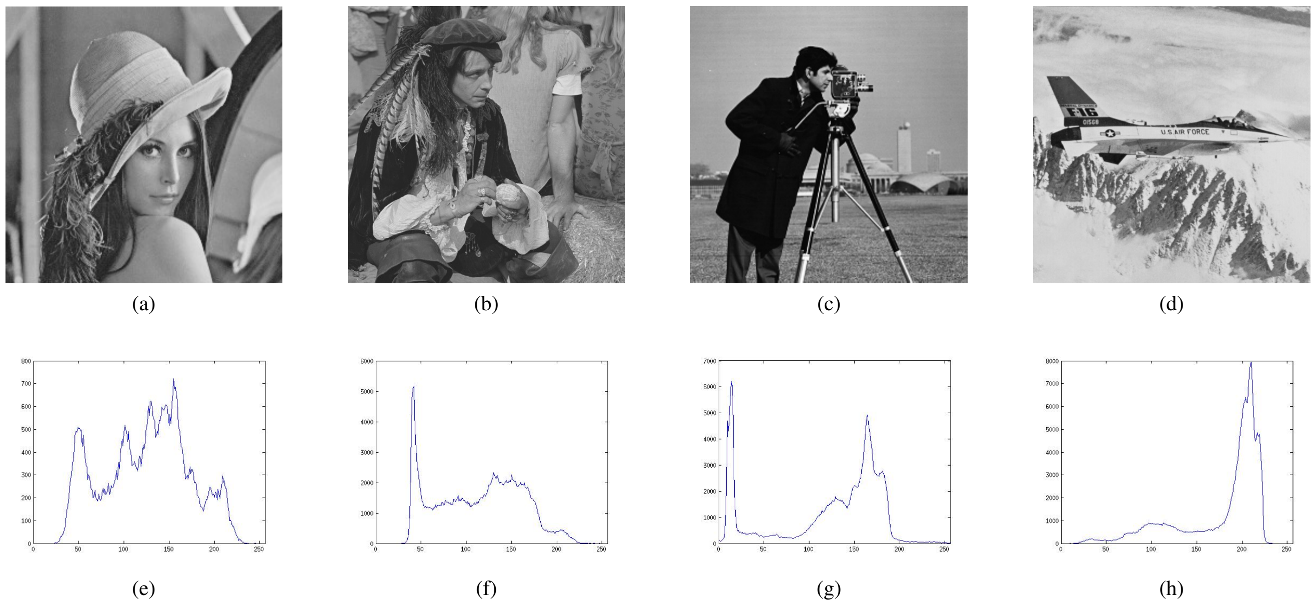


Fig. 1. Test images and their corresponding histograms. (a) Lena, (b) Pirate, (c) Cameraman, (d) Jetplane, (e) Lena histogram, (f) Pirate histogram, (g) Cameraman histogram, (h) Jetplane histogram.

TABLE I
COMPARISON OF BEST MEAN OBJECTIVE VALUES, WITH MEAN CPU TIMES COMPUTED BY DE, JDE, L-SHADE, JDE$_{Kent}$, JDE$_{Log}$, AND JDE$_{Tent}$ USING KAPUR'S ENTROPY.

Test image	M	Mean objective values						Mean CPU time (s)					
		DE	jDE	L-Shade	jDE$_{Kent}$	jDE$_{Log}$	jDE$_{Tent}$	DE	jDE	L-Shade	jDE$_{Kent}$	jDE$_{Log}$	jDE$_{Tent}$
Cameraman	2	**12.2865**	**12.2865**	**12.2865**	**12.2865**	12.2864	12.2755	0.007933	**0.005533**	0.09727	0.0165	0.006333	0.01353
	4	**18.5566**	18.5411	**18.5566**	**18.5566**	18.5548	18.5486	0.03623	**0.01093**	0.4641	0.0417	0.01267	0.05563
	6	**24.0475**	24.0162	24.0284	24.0349	24.0145	23.9877	0.1361	**0.03217**	0.541	0.07487	0.0333	0.1356
	8	29.0174	28.9506	28.9805	**29.0441**	28.985	28.9182	0.2358	**0.0387**	0.7035	0.1397	0.04107	0.1478
	10	33.4566	33.5101	33.3654	**33.536**	33.4688	33.4034	0.2194	0.05507	0.6755	0.2107	**0.0446**	0.1628
	12	37.4313	**37.6009**	37.3881	37.5821	37.5803	37.4017	0.216	0.0728	0.6649	0.2524	**0.0548**	0.1776
Jetplane	2	12.2422	12.2417	**12.2427**	**12.2427**	12.2408	12.2382	0.007133	0.006133	0.1204	0.0129	**0.0053**	0.0108
	4	**18.3397**	18.339	18.3395	**18.3397**	18.3388	18.3316	0.0363	**0.01413**	0.6295	0.0354	0.015	0.05303
	6	**23.3563**	23.3286	23.3487	23.3418	23.3165	23.3097	0.1857	0.03723	0.6496	0.0763	**0.02377**	0.1046
	8	27.8426	27.8354	27.8272	**27.8685**	27.8153	27.7896	0.2128	0.03977	0.6481	0.1268	**0.03267**	0.1416
	10	31.7967	31.8122	31.7673	**31.8689**	31.8018	31.727	0.2071	0.0503	0.6114	0.1928	**0.03727**	0.1624
	12	35.3962	35.4535	35.3617	**35.4877**	35.4652	35.3255	0.2022	0.08537	0.5963	0.222	**0.0658**	0.1867
Lena	2	**12.344**	**12.344**	**12.344**	**12.344**	**12.344**	12.3425	0.002633	**0.001767**	0.02203	0.004667	0.001933	0.003067
	4	18.0051	17.9939	**18.006**	18.0035	17.9955	17.9893	0.02583	0.006567	0.1458	0.01547	**0.005067**	0.02303
	6	22.9775	22.9617	22.971	**22.9785**	22.9546	22.9478	0.0838	0.008833	0.1691	0.03517	**0.008033**	0.04513
	8	27.2804	27.258	27.2612	**27.3103**	27.2459	27.2459	0.09947	0.0167	0.1821	0.06113	**0.0119**	0.0643
	10	31.2279	31.236	31.1904	**31.3018**	31.2388	31.1669	0.0947	0.0204	0.1929	0.0874	**0.01807**	0.07307
	12	34.8056	34.9039	34.7315	**34.9471**	34.8765	34.7849	0.09707	0.03163	0.2	0.09977	**0.02593**	0.0824
Pirate	2	**12.0033**	**12.0033**	**12.0033**	**12.0033**	**12.0033**	12	0.006967	**0.0053**	0.1026	0.01483	0.0056	0.01083
	4	**17.6585**	17.655	17.6584	17.6583	17.655	17.6486	0.03893	**0.01183**	0.538	0.04007	0.0166	0.05483
	6	**22.396**	22.3861	22.3896	22.3922	22.37	22.3578	0.201	**0.02797**	0.5639	0.0841	0.02853	0.09927
	8	26.8266	26.7759	26.7874	**26.8518**	26.7625	26.7528	0.2245	0.03843	0.5686	0.1263	**0.0318**	0.1367
	10	30.7942	30.8072	30.7437	**30.8728**	30.788	30.6882	0.2144	0.05703	0.5842	0.2012	**0.0499**	0.1585
	12	34.3153	**34.4399**	34.2567	34.4061	34.3881	34.2478	0.2047	0.07237	0.5847	0.2166	**0.0615**	0.1813

and C_2 are constants, which are included to avoid instability when $\mu_I^2 + \mu_J^2$ is close to zero ($C_1 = 6.5025$, $C_2 = 58.5225$).

As mentioned in Section V, the Kapur's entropy was utilized as the objective function. In this section, we present the numerical results of the optimal thresholding and also report the segmentation qualities of the found thresholds using the variables $PSNR$ and $SSIM$. The results are collated in Tables I, II and III, while Figures 2, 3, 4 and 5 represent segmented images by applying the best found thresholds. The best results in Tables are marked in bold text.

The results in Table I summarize the best mean objective values obtained by each algorithm and also the average CPU time is also reported in seconds. Based on the mean objective values, the jDE$_{Kent}$ obtained the best results, since it achieved the highest values in 17 instances. Interestingly, the fastest method was the jDE$_{Log}$, despite achieving very poor results regarding the mean objective value. The DE, jDE, and L-Shade achieved fairly similar results, with the original DE obtaining the best results in 9 instances.

Table II provides a very interesting analysis. Despite the jDE$_{Kent}$ being the best method based on mean objective values, the jDE$_{Log}$ is best based on mean PSNR, achieving

TABLE II

COMPARISON OF BEST AND MEAN PSNR VALUES COMPUTED BY DE, JDE, L-SHADE, JDE$_{Kent}$, JDE$_{Log}$, AND JDE$_{Tent}$ USING KAPUR'S ENTROPY.

Test image	M	Mean PSNR values						Best PSNR values					
		DE	jDE	L-Shade	jDE$_{Kent}$	jDE$_{Log}$	jDE$_{Tent}$	DE	jDE	L-Shade	jDE$_{Kent}$	jDE$_{Log}$	jDE$_{Tent}$
Cameraman	2	12.3596	12.3596	12.3596	12.3596	12.3596	**12.6524**	12.359600	12.359600	12.359600	12.359600	12.359600	**12.733200**
	4	18.7247	18.7247	18.7247	18.7247	18.7247	**18.9743**	18.993000	**19.251900**	18.734500	18.993000	18.993000	19.099600
	6	21.0304	21.0304	20.9466	21.0304	20.9035	**21.0314**	21.150100	21.824500	21.342800	21.801400	22.023700	**22.363300**
	8	23.3645	25.7138	22.7434	23.7674	**25.7674**	23.1902	25.336800	25.736300	25.525800	25.664000	25.767400	**25.873900**
	10	24.2294	26.9639	26.8608	26.9017	**27.1849**	26.9168	27.318100	27.858600	27.178000	**27.975500**	27.855500	27.770400
	12	28.257	28.672	28.0585	28.6593	**28.7419**	28.1981	29.116900	**29.354200**	28.974600	28.894900	28.941500	29.032900
Jetplane	2	13.6047	13.6047	13.6047	13.6047	**13.9169**	13.6047	**13.916900**	**13.916900**	13.604700	13.604700	**13.916900**	**13.916900**
	4	15.486	**15.4888**	15.486	15.486	15.486	15.4247	15.486000	15.489600	15.489600	15.486000	15.491900	**15.695900**
	6	15.8073	16.0383	15.8991	15.9056	**16.8276**	16.0353	16.660600	16.668300	16.114400	16.259800	17.687600	**21.016700**
	8	17.83	17.129	17.3395	18.1003	**19.3374**	19.0139	19.342000	20.049700	19.030700	18.698000	20.945900	**22.395700**
	10	19.161	20.713	17.4116	20.696	**22.7395**	18.7546	22.356500	**23.500100**	23.451000	21.237900	23.426900	23.474800
	12	20.7842	21.3217	20.3233	20.8084	**30.3105**	20.7469	24.316300	25.508200	24.250200	25.145300	**30.310500**	25.937800
Lena	2	**15.3922**	**15.3922**	**15.3922**	**15.3922**	**15.3922**	**15.3922**	15.392200	15.392200	15.392200	15.392200	15.426000	**15.527700**
	4	19.2781	**19.9098**	19.2742	19.2781	19.7086	19.3526	19.909800	**19.948800**	19.339400	19.909800	19.909800	19.909800
	6	22.691	22.9464	22.5986	22.691	**23.5434**	22.6	22.879300	23.525600	22.878600	22.869300	23.551300	**23.734400**
	8	26.0959	25.2248	24.4398	**26.4532**	23.8557	25.1451	**26.578700**	26.467500	26.380700	26.461000	26.457600	26.460600
	10	27.6907	27.8523	**28.4177**	27.7279	27.0339	28.3116	28.448900	28.488100	28.557100	**28.573700**	28.507700	28.311600
	12	29.2924	**30.0984**	29.4302	29.4418	28.2879	29.1704	29.957100	**30.098400**	30.006700	29.882000	30.077300	30.094900
Pirate	2	**15.0289**	**15.0289**	**15.0289**	**15.0289**	**15.0289**	14.8072	15.028900	15.028900	15.028900	15.028900	15.028900	**15.256900**
	4	20.2454	20.2454	**20.2514**	20.2454	20.2454	20.2454	20.245400	20.346200	20.251400	20.245400	20.346200	**20.433600**
	6	23.3456	23.3388	23.0403	23.2948	**23.499**	23.4612	23.526900	23.526900	23.527400	23.526900	23.582600	**23.774500**
	8	25.4394	26.1338	25.3582	26.1227	**26.4099**	25.704	26.026800	26.544300	25.816700	26.130400	26.498800	**26.613700**
	10	27.3313	28.2588	28.1003	28.1812	27.6326	**28.5772**	28.317300	28.642100	28.273800	28.398000	**28.701900**	28.577200
	12	29.2492	**30.027**	29.8518	29.8984	29.2036	29.3289	30.165600	30.259400	30.327900	**30.475300**	30.417700	30.003600

TABLE III

COMPARISON OF BEST AND MEAN SSIM VALUES COMPUTED BY DE, JDE, L-SHADE, JDE$_{Kent}$, JDE$_{Log}$, AND JDE$_{Tent}$ USING KAPUR'S ENTROPY.

Test image	M	Mean SSIM values						Best SSIM values					
		DE	jDE	L-Shade	jDE$_{Kent}$	jDE$_{Log}$	jDE$_{Tent}$	DE	jDE	L-Shade	jDE$_{Kent}$	jDE$_{Log}$	jDE$_{Tent}$
Cameraman	2	0.631909	0.631909	0.631909	0.631909	0.631909	**0.638434**	0.631909	0.631909	0.631909	0.631909	0.631909	**0.639749**
	4	0.744457	0.744457	0.744457	0.744457	0.744457	**0.744746**	0.746217	0.755225	0.745266	0.746217	0.751425	**0.758728**
	6	0.806079	0.806079	**0.815235**	0.806079	0.802476	0.805833	0.810285	0.824556	0.815235	0.821852	**0.827346**	0.826608
	8	0.84577	0.847675	0.85041	0.847332	**0.854299**	0.842927	0.856523	0.859208	0.860297	0.851170	0.857465	**0.863194**
	10	0.857823	0.870537	0.857136	0.8715	**0.873763**	0.870142	0.880227	0.885122	0.871042	**0.886116**	0.884030	0.882035
	12	0.884603	0.893052	0.874679	0.894005	**0.895747**	0.885887	**0.900261**	0.899272	0.898976	0.895360	0.897160	0.895555
Jetplane	2	0.757859	0.757859	0.757859	0.757859	**0.763005**	0.757859	0.763005	0.763005	0.757859	0.757859	0.763005	**0.763637**
	4	0.798905	0.799323	0.798905	0.798905	0.798905	**0.80362**	0.798905	0.803837	0.803518	0.798905	0.803620	**0.807261**
	6	**0.804604**	0.804388	0.800967	0.801478	0.770107	0.803924	0.809413	0.808360	0.813889	0.804388	0.809680	**0.814683**
	8	0.761106	**0.77537**	0.76875	0.760763	0.766419	0.76379	0.789023	0.793276	**0.812467**	0.782795	0.806104	0.803115
	10	0.776587	0.777575	0.77879	0.776253	**0.812936**	0.766437	0.801461	**0.835434**	0.834710	0.780978	0.833285	0.834900
	12	0.783423	0.786605	0.784624	0.785251	**0.910886**	0.779839	0.858495	0.872925	0.854468	0.854471	**0.910886**	0.863033
Lena	2	**0.646656**	**0.646656**	**0.646656**	**0.646656**	**0.646656**	**0.646656**	0.646656	0.646656	0.646656	0.646656	0.646656	**0.657604**
	4	0.755555	0.7591	0.75686	0.755555	**0.76906**	0.753537	0.759100	**0.772368**	0.756860	0.771363	0.772088	0.772100
	6	0.8239	0.818639	0.825057	0.8239	0.81104	**0.825623**	0.824986	**0.826188**	0.825162	0.824757	0.824757	0.825719
	8	0.857382	0.86005	0.853231	**0.860341**	0.853586	0.851419	0.865187	**0.866019**	0.862370	0.865887	0.864937	0.865878
	10	0.880768	0.879695	**0.893952**	0.879189	0.876722	0.887371	0.893894	0.895934	0.894688	**0.896568**	0.895850	0.896233
	12	0.905753	**0.917113**	0.904617	0.90681	0.89377	0.894826	0.914920	**0.917388**	0.915304	0.914075	0.915690	0.915304
Pirate	2	**0.596805**	**0.596805**	**0.596805**	**0.596805**	**0.596805**	0.595282	0.596805	0.596805	0.596805	0.596805	0.596805	**0.599208**
	4	0.747946	0.747946	**0.747962**	0.747946	0.747946	0.747946	0.747946	0.750798	0.748011	0.748085	0.750798	**0.750832**
	6	0.810056	0.807803	0.808664	**0.827037**	0.816087	0.810347	0.812245	0.827400	0.815756	**0.828531**	0.827606	0.826613
	8	0.851647	0.867826	0.850671	0.868329	**0.869785**	0.853071	0.857583	**0.870851**	0.869903	0.868436	0.870474	0.869505
	10	0.886486	0.895181	0.892986	0.894169	0.892394	**0.900463**	0.896369	0.902054	0.895802	0.897103	**0.903646**	0.900463
	12	0.910983	0.918543	0.917691	**0.919516**	0.909938	0.910329	0.921941	0.921254	0.918639	**0.924161**	0.922359	0.922844

the highest score in 13 instances. On the other hand, when comparing the best PSNR values, interestingly the jDE$_{Tent}$ was the best.

The images were also compared on the basis of the $SSIM$ performance metric. The results of this comparison are gathered in Table III. The performance of the algorithms, based on $SSIM$ are very similar to those of $PSNR$. Based on the mean $SSIM$, the best was again the jDE$_{Log}$, while the jDE$_{Tent}$ obtained the best result when considering the best $SSIM$.

Friedman tests [9] were conducted in order to estimate the quality of the results obtained by various DE algorithms for the gray-level image segmentation statistically. The Friedman test is a two-way analysis of variances by ranks, where the statistic test is calculated and converted to ranks in the first step. The post-hoc tests are conducted using the calculated ranks in the second step. Here, a low value of rank means a better algorithm [6]. The second step is performed only if a null hypothesis of Friedman test is rejected. Note, the null hypothesis states that medians between the ranks of all algorithms are equal.

According to Demšar [5], the Friedman test is a more safe and robust non-parametric test for the comparisons of more algorithms over multiple classifiers (also datasets) that, together with the corresponding Nemenyi post-hoc test enables a neat presentation of statistical results [13]. The main drawback of the Friedman test is that it makes the whole multiple comparisons over datasets and it is, therefore, unable to establish proper comparisons between some of the algorithms considered [6]. Consequently, a Wilcoxon two paired non-parametric test was applied as a post-hoc test after determining

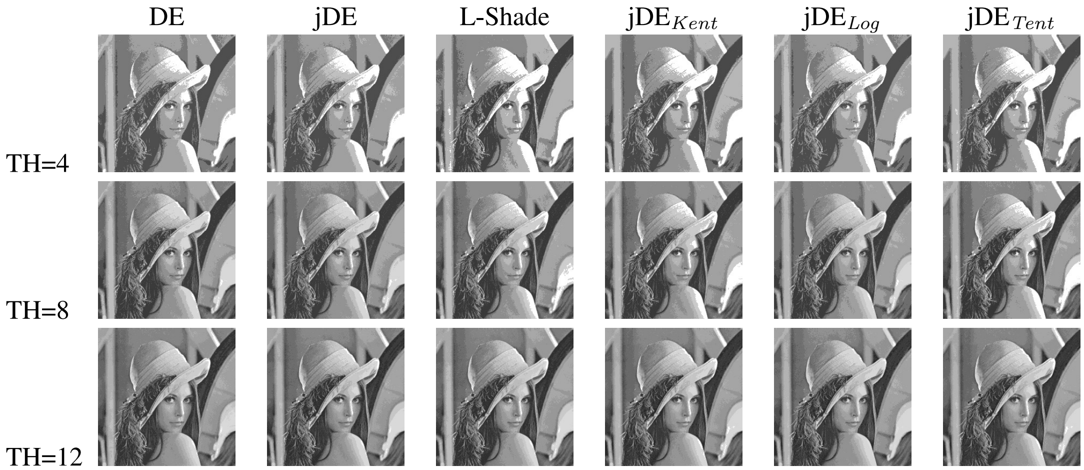


Fig. 2. Image Lena segmented into 4, 8, and 12 levels.

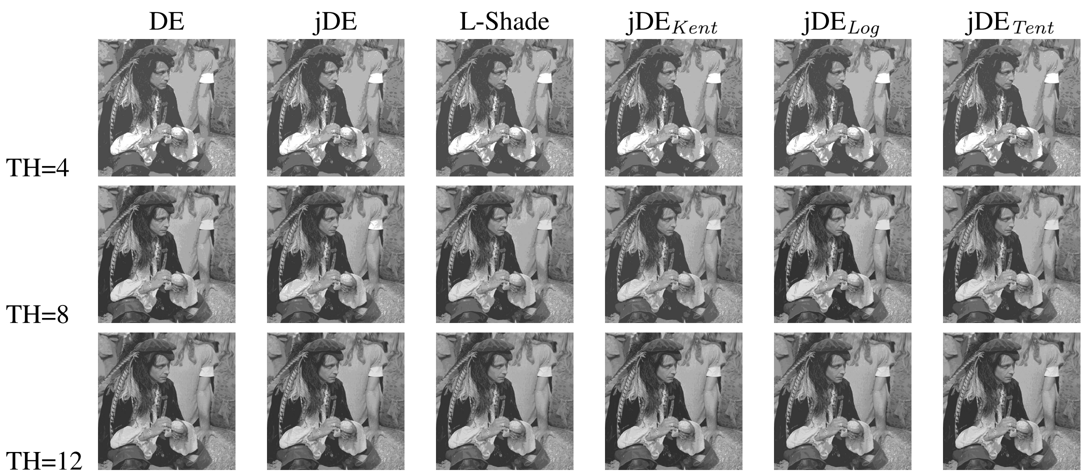


Fig. 3. Image Pirate segmented into 4, 8, and 12 levels.

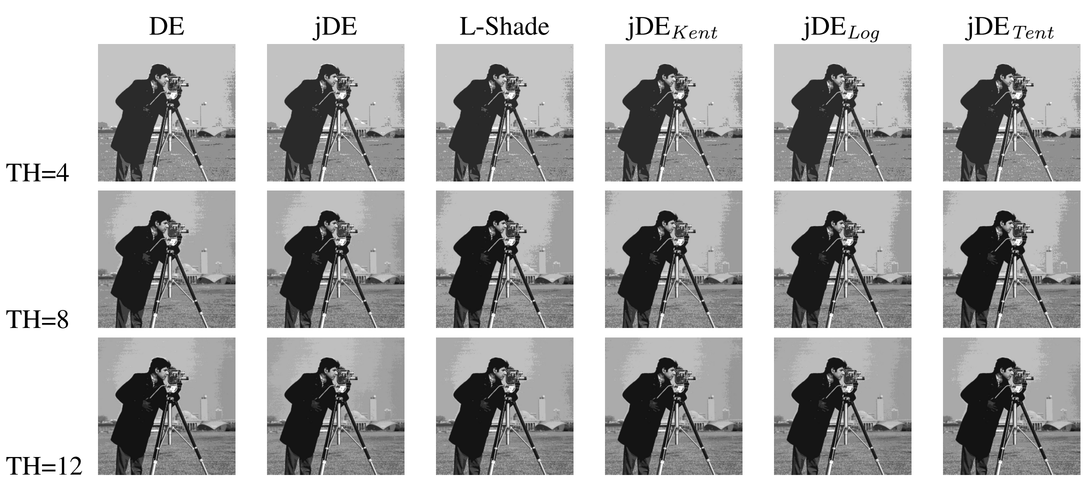


Fig. 4. Image Cameraman segmented into 4, 8, and 12 levels.

the control method (i.e., the algorithm with the lowest rank) by using the Friedman test. On the other hand, the Nemenyi test is very conservative and it may not find any difference in most of the experimentations [10]. Therefore, the Nemenyi test is used for graphical presentation of the results, while the Wilcoxon test shows which of the algorithms in test are more powerful. Both tests were conducted using the significance level 0.05 in this study.

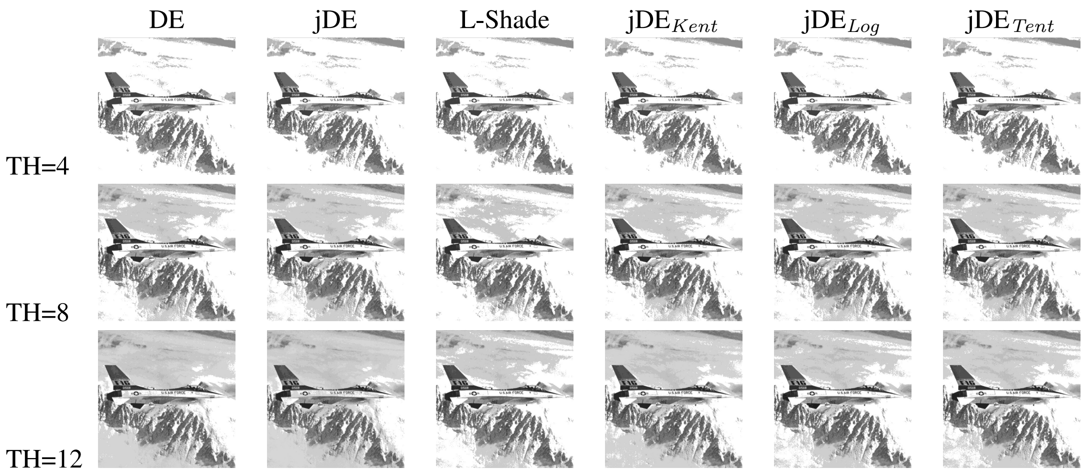
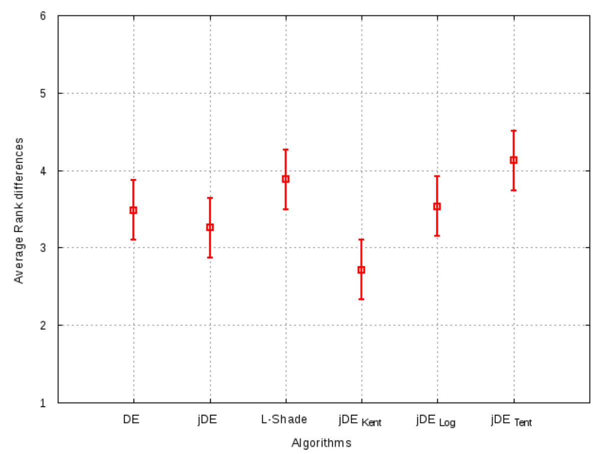


Fig. 5. Image Jetplane segmented into 4, 8, and 12 levels.

Algorithms	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
DE	3.48	[3.10,3.86]		0.05167	-
jDE	3.26	[2.88,3.64]		$\ll 0.05$	+
L-Shade	3.88	[3.50,4.26]	†	$\ll 0.05$	+
jDE$_{Kent}$	2.71	[2.33,3.10]	‡	∞	+
jDE$_{Log}$	3.53	[3.15,3.92]	†	$\ll 0.05$	+
jDE$_{Tent}$	4.13	[3.74,4.51]	†	$\ll 0.05$	+

(a) Numerical results of the Friedman and Wilcoxon tests.



(b) Graphical representation of ranks and critical distances

Fig. 6. Statistical analysis of DE algorithms for image segmentation

The results of the statistical tests are illustrated in Fig. 6, which is divided into two diagrams. The first diagram represents a table with the numerical results of three statistical tests, i.e., the Friedman non-parametric test, together with the Nemenyi and Wilcoxon non-parametric tests. The values were obtained by comparing the best fitness values for each of the observed axes together with their corresponding mean values. As a result, each observed algorithm (i.e., classifier) consists of $4 \times 6 \times 4 = 96$ values (i.e., 4 images $\times$ 6 different threshold levels $\times$ 4 statistical measures: the best, mean, worst, and standard deviation, were considered). The second diagram illustrates the results of the Nemenyi post-hoc statistical test graphically.

As can be seen from the Fig. 6(a), the jDE$_{Kent}$ algorithm achieved the best results due to the minimum rank value according to the Friedman non-parametric test. Therefore, this algorithm represents the control method with which the other algorithms were compared. The control method is denoted by the sign '‡' in the table. The interval in the column 'CD' by the Nemenyi test denotes the confidence interval according to which the significant difference can be determined between two algorithms. In line with this, two algorithms are significantly different, if their critical differences (CD) do not overlap. The significant differences are denoted in the table by the sign '†'.

Let us notice that both post-hoc statistical tests return the same results, where the jDE$_{Kent}$ algorithm (i.e., the control method) outperformed the results of the algorithms, like jDE, L-Shade, jDE$_{Log}$ and jDE$_{Tent}$ significantly. Interestingly, the difference between jDE$_{Kent}$ and DE is not significant.

VII. Conclusion

A study of the impact of different chaotic maps, embedded into a self-adaptive differential evolution for the purpose of image segmentation was performed in this paper. The used objective was the Kapur entropy, which works by maximizing the entropy of different regions in the input image. Three chaotic maps were considered, namely the Kent, Logistic and Tent maps commonly found in literature. The applied chaotic methods were compared to DE, jDE, and L-Shade, tested on four different images, which are usually found in computer vision benchmarks. The comparison was made based on the objective value, and two image quality performance metrics, such as $PSNR$ and $SSIM$. The results show, that the chaotic embedded methods performed better. The best performing map was the Tent map, while the fastest convergence was obtained with the Logistic map.

For future work, we plan to study other chaotic maps, while also considering other objective functions, such as maximizing between-class variance or various entropy measures like Tsallis or Renyi entropy.

ACKNOWLEDGMENT

REFERENCES

[1] Adis Alihodzic and Milan Tuba. Improved bat algorithm applied to multilevel image thresholding. *The Scientific World Journal*, 2014, 2014.

[2] AK Bhandari, A Kumar, and GK Singh. Tsallis entropy based multilevel thresholding for colored satellite image segmentation using evolutionary algorithms. *Expert Systems with Applications*, 42(22):8707–8730, 2015.

[3] Janez Brest, Sašo Greiner, Borko Boskovic, Marjan Mernik, and Viljem Zumer. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation*, 10(6):646–657, 2006.

[4] Erik Cuevas, Daniel Zaldivar, and Marco Prez-Cisneros. A novel multi-threshold segmentation approach based on differential evolution optimization. *Expert Systems with Applications*, 37(7):5265 – 5271, 2010.

[5] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[6] Joaqun Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[7] David P Feldman. *Chaos and fractals: an elementary introduction*. Oxford University Press, 2012.

[8] Iztok Fister Jr., Xin-She Yang, Janez Brest, Dušan Fister, and Iztok Fister. Analysis of randomisation methods in swarm intelligence. *International journal of bio-inspired computation*, 7(1):36–49, 2015.

[9] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11:86–92, 1940.

[10] Salvador García and Francisco Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, pages 2677–2694, 2008.

[11] Ming-Huwi Horng. Multilevel thresholding selection based on the artificial bee colony algorithm for image segmentation. *Expert Systems with Applications*, 38(11):13785–13791, 2011.

[12] Jagat Narain Kapur, Prasanna K Sahoo, and Andrew KC Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer vision, graphics, and image processing*, 29(3):273–285, 1985.

[13] Peter B. Nemenyi. *Distribution-free multiple comparisons*. Princeton University, 1963.

[14] Edward Ott. *Chaos in Dynamical Systems*. Cambridge University Press, 2002.

[15] Michal Pluhacek, Roman Senkerik, and Ivan Zelinka. Particle swarm optimization algorithm driven by multichaotic number generator. *Soft Computing*, 18(4):631–639, 2014.

[16] Soham Sarkar, Swagatam Das, and Sheli Sinha Chaudhuri. *Multilevel Image Thresholding Based on Tsallis Entropy and Differential Evolution*, pages 17–24. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[17] Soham Sarkar, Gyana Ranjan Patra, and Swagatam Das. *A Differential Evolution Based Approach for Multilevel Image Segmentation Using Minimum Cross Entropy Thresholding*, pages 51–58. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[18] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[19] Shilpa Suresh and Shyam Lal. An efficient cuckoo search algorithm based multilevel thresholding for segmentation of satellite images using different objective functions. *Expert Syst. Appl.*, 58(C):184–209, October 2016.

[20] Ryoji Tanabe and Alex S Fukunaga. Improving the search performance of shade using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1658–1665. IEEE, 2014.

[21] USC Viterbi. The usc-sipi image database, 2016.

[22] Aleš Zamuda and Janez Brest. Self-adaptive control parameters randomization frequency and propagations in differential evolution. *Swarm and Evolutionary Computation*, 25:72–99, 2015.