Multilevel Image Segmentation based on Fractional-Order Darwinian Particle Swarm Optimization

Pedram Ghamisi[1], *Student Member, IEEE,* Micael S. Couceiro[2,4], *Student Member, IEEE*
Fernando M.L. Martins[3,4] and Jon Atli Benediktsson[1], *IEEE Fellow*

[1]Faculty of Electrical and Computer Engineering, University of Iceland, Hjardarhaga 2-6, 101 Reykjavik, Iceland,

[2]Institute of Systems and Robotics – DEEC, Polo II, Pinhal de Marrocos, University of Coimbra, Coimbra, Portugal

[3]Instituto de Telecomunicações (Covilhã), Coimbra College of Education, Coimbra, Portugal

[4]RoboCorp at the Engineering Institute of Coimbra, Polytechnic Institute of Coimbra, Rua Pedro Nunes, Coimbra, Portugal

Abstract: Hyperspectral remote sensing images contain hundreds of data channels. Due to the high dimensionality of the hyperspectral data, it is difficult to design accurate and efficient image segmentation algorithms for such imagery. In this paper, a new multilevel thresholding method is introduced for the segmentation of hyperspectral and multispectral images. The new method is based on Fractional-Order Darwinian Particle Swarm Optimization (*FODPSO*) which exploits the many swarms of test solutions that may exist at any time. In addition, the concept of fractional derivative is used to control the convergence rate of particles. In this work, the so-called Otsu problem is solved for each channel of the multispectral and hyperspectral data. Therefore, the problem of n-level thresholding is reduced to an optimization problem in order to search for the thresholds that maximizes the between-class variance. Experimental results are favorable for the *FODPSO* when compared to other bio-inspired methods for multi-level segmentation of multispectral and hyperspectral images. The *FODPSO* presents a statistically significant improvement in terms of both *CPU* time and fitness value, i.e., the approach is able to find the optimal set of thresholds with a larger between-class variance in less computational time than the other approaches. In addition, a new classification approach based on Support Vector Machine *(SVM)* and *FODPSO* is introduced in this paper. Results confirm that the new segmentation method is able to improve upon results obtained with the standard *SVM* in terms of classification accuracies.

Keywords: multilevel segmentation; swarm optimization; image processing, classification

I. INTRODUCTION

Image segmentation is regarded as the process of partitioning a digital image into multiple regions or objects. In other words, in image segmentation a label is assigned to each pixel in the image such that pixels with the same label share certain visual characteristics [1]. These objects provide more information than individual pixels since the interpretation of images based on objects is more meaningful than the interpretation based on individual pixels only. Image segmentation is considered as an important task in the analysis, interpretation and understanding of images and is also widely used for image processing purposes such as classification and object recognition [1, 2].

Image segmentation plays a key role in the field of remote sensing image analysis. For example, in order to improve classification results, the integration of classification and segmentation steps has recently been taken into account [3, 4]. In such cases, the decision to assign a pixel to a specific class is simultaneously based on the feature vector of this pixel and some additional information derived from the segmentation step. To make this approach effective, an accurate segmentation of the image is needed. A few methods for segmentation of multispectral and hyperspectral images have been introduced in the literature. Some of these methods are based on region merging techniques, in which neighboring image segments are merged with each other based on their homogeneity. For example, the multiresolution segmentation method in eCognition software uses this type of approach [5]. Tilton proposed a hierarchical segmentation algorithm [6], which alternately performs region growing and spectral clustering. There is an extensive number of segmentation methods that have been proposed in the literature that exploit mathematical morphology approaches [7-15], for segmentation of multispectral and hyperspectral images.

Image segmentation can be classified into four specific types including histogram thresholding based methods, texture analysis based methods, clustering based methods and region based split and merging methods. *Thresholding* is one of the most commonly used methods for the segmentation of images into two or more clusters [16]. Thresholding techniques can be divided into two different

types: Optimal thresholding methods [17-21] and property based thresholding methods [22-24]. Algorithms in the former group search for the optimal thresholds which make the thresholded classes on the histogram reach the desired characteristics. Usually, thresholds are selected by optimizing an objective function. The latter group detects the thresholds by measuring some selected property of the histogram. Property-based thresholding methods are fast, which makes them suitable for multilevel thresholding. However, the number of thresholds for these approaches is hard to determine and needs to be specified in advance.

Many algorithms have been proposed in literature to address the issue of optimal thresholding (*e.g.*, [17, 19-20, 24-28]). While several research papers address bi-level thresholding, others have considered the multilevel problem. Bi-level thresholding is reduced to an optimization problem to determine the threshold t that maximizes the σ_B^2 (between − class variance) and minimizes σ_W^2 (within − class variance) [16]. For two level thresholding, the problem is solved by finding the value T^* which satisfies max(σ_B^2 (T^*)) where $0 \leq T^* < L$ and L is the maximum intensity value. This problem could be extended to n-level thresholding through satisfying max σ_B^2 ($T^*_1, T^*_2, \ldots, T^*_{n-1}$) where $0 \leq T^*_1 < T^*_2 < \ldots < T^*_{n-1} < L$. One way for finding the optimal set of thresholds is by using exhaustive search. A commonly used exhaustive search is based on the Otsu criterion [19]. That approach is easy to implement, but it has the disadvantage that it is computationally expensive. Exhaustive search for *n - 1* optimal thresholds involves evaluations of fitness of $n(L-n+1)^{n-1}$ combinations of thresholds [16]. Therefore, that method is not suitable from a computational cost point of view. The task of determining *n - 1* optimal thresholds for *n*-level image thresholding could be formulated as a multidimensional optimization problem. To solve such a task, several biologically inspired algorithms have been explored in image segmentation [16, 29-32]. Bio-inspired algorithms have been used in situations where conventional optimization techniques cannot find a satisfactory solution or they take too much time to find it, e.g., when the function to be optimized is discontinuous, non-differentiable, and/or presents too many nonlinearly related parameters [33].

One of the best known bio-inspired algorithms is Particle Swarm Optimization (*PSO*) [34]. The *PSO* consists of a number of particles that collectively move in the search space (*e.g.*, pixels of the image) in search of the global optimum (*e.g.*, maximizing the between-class variance of the distribution of intensity levels in the given image). However, a general problem with the *PSO* and similar optimization algorithms is that they may get trapped in local optimum points, and the algorithm may work in some problems but fail in others [35]. To overcome such a problem, Tillett *et al.* [36] presented the Darwinian *PSO* (*DPSO*). In the *DPSO*, multiple swarms of test solutions performing just like an ordinary *PSO* may exist at any time with rules governing the collection of swarms that are designed to simulate natural selection. More recently, Couceiro *et al.* [35] further extended the *DPSO* using fractional calculus to control the convergence rate of the algorithm. In [35], fractional-order *DPSO* (*FODPSO*) was successfully compared to both the fractional-order *PSO* (*FOPSO*) from Pires *et al.* [37] and the traditional *DPSO*.

The main goal of this study is to propose a computationally efficient bio-inspired segmentation method, which is robust for partitioning remote sensing images into multiple regions. For this purpose, a new method for segmentation of multispectral and hyperspectral images based on the *FODPSO* is proposed. To demonstrate the performance of this new method, a methodical and statistic comparison with two other methods for thresholding segmentation of images, namely the well-known *PSO* and the *DPSO*, is carried out. In summary, the main contributions of the paper are as follows:

i) Formal presentation of the *FODPSO* algorithm for image segmentation;
ii) Evaluation of this novel algorithm using more complex data sets (*i.e.*, multispectral/hyperspectral) and comparison with other thresholding based segmentation methods;
iii) Proposition of a novel classification approach based on the concept of the new segmentation method to improve the classification accuracy of the traditional Support Vector Machine (*SVM*) method.

It should be noted that this is the first time that the concept of *FODPSO* is used in remote sensing, thus showing the potential of its use in efficient image segmentation to determine broad groups of objects. The current paper partially builds on [34] with an important study on how *FODPSO* performs for remote sensing images while the segmentation level for such images are changed. Moreover, a deep statistical analysis is conducted so as to further sustain the proposed approach when compared to others. Many problems in remote sensing have been solved by considering optimization methods such as Genetic Algorithm (*GA)* and *PSO*. Therefore, this paper introduces a very powerful optimization method, both in terms of speed and optimal convergence, which can be considered for a wide variety of problems in remote sensing. Some optimization methods are fast but not efficient (for finding the global optimum) and vice-versa. It has been recently proved in [35] for benchmarking optimization problems that the *FODPSO* is faster than the *PSO* (the most well-known optimization algorithm in terms of speed) and more efficient than the *DPSO* (in order to find the global optimum while avoiding local optima). Therefore, applying the *FODPSO* to the segmentation of images may allow achieving both vital important goals at once. More specifically, due to its convergence speed, this optimization method may present itself as a potential solution to a wide variety of complex problems in remote sensing such as hyperspectral image analysis - a problem that many researchers are struggling with since *hyperspectral* images in remote sensing are very volumetric.

Bearing these ideas in mind, the problem formulation of image *n*-level thresholding is presented in the following subsections. Section II presents a brief review of *PSO* and *DPSO* algorithms and focuses on their strengths and weaknesses, thus paving the way for a detailed description on the method that is proposed in the paper. In Section III, two different remote sensing data sets are considered and the performance of the three different methods is compared. In Section IV, the proposed segmentation approach is extended and applied for classification. Finally, the main conclusions are outlined in Section IV.

II. METHODOLOGY

Multilevel segmentation techniques provide an efficient way to perform image analysis. However, the automatic selection of a robust optimum *n*-level threshold remains a challenge in segmentation of remote sensing images. Below a more precise formulation of the problem is introduced along with the basic notation used in the paper.

A. Problem Formulation

Let there be L intensity levels in each component, *e.g.*, three color components for *RGB* images, of a given image and these levels are in the range $\{0, 1, 2, \ldots, L\text{-}1\}$. Then one can define:

$$p_i^C = \frac{h_i^C}{N},\ \sum_{i=0}^{L-1} p_i^C = 1, \tag{1}$$

where i represents a specific intensity level, *i.e.*, $0 \le i \le L$-1, C represents the component of the image, *e.g.*, $C = \{R,\ G,\ B\}$ for *RGB* images, N represents the total number of pixels in the image and h_i^C denotes the number of pixels for the corresponding intensity level i in the component C. In other words, h_i^C represents an image histogram for each component C, which can be normalized and regarded as the probability distribution p_i^C. The total mean (*i.e.*, combined mean) of each component of the image can be easily calculated as:

$$\mu_T^C = \sum_{i=0}^{L-1} i p_i^C. \tag{2}$$

The n-level thresholding presents $n-1$ threshold levels $t_j^C, j = 1, \ldots, n-1$, and the operation is performed as:

$$F^C(x,y) = \begin{cases} 0, & f^C(x,y) \le t_1^C \\ \frac{1}{2}(t_1^C + t_2^C), & t_1^C < f^C(x,y) \le t_2^C \\ \vdots & \\ \frac{1}{2}(t_{n-2}^C + t_{n-1}^C), & t_{n-2}^C < f^C(x,y) \le t_{n-1}^C \\ L-1, & f^C(x,y) > t_{n-1}^C \end{cases}. \tag{3}$$

where x and y are the width (W) and height (H) pixel of the image of size $H \times W$ denoted by $f^C(x,y)$ with L intensity levels for each component. In this situation, the pixels of a given image will be divided into n classes $D_1^C, \ldots, D_n^C$, which may represent multiple objects or even specific features for such objects (*e.g.*, topological features). The probabilities of occurrence w_j^C of classes $D_1^C, \ldots, D_n^C$ are given by

$$w_j^C = \begin{cases} \sum_{i=0}^{t_j^C} p_i^C, & j = 1 \\ \sum_{i=t_{j-1}^C+1}^{t_j^C} p_i^C, & 1 < j < n, \\ \sum_{i=t_{j-1}^C+1}^{L-1} p_i^C, & j = n. \end{cases} \tag{4}$$

The mean of each class μ_j^C can then be calculated as

$$\mu_j^C = \begin{cases} \sum_{i=0}^{t_j^C} \frac{i p_i^C}{w_j^C}, & j = 1 \\ \sum_{i=t_{j-1}^C+1}^{t_j^C} \frac{i p_i^C}{w_j^C}, & 1 < j < n. \\ \sum_{i=t_{j-1}^C+1}^{L-1} \frac{i p_i^C}{w_j^C}, & j = n. \end{cases} \tag{5}$$

The simplest and computationally most efficient method of obtaining the optimal threshold is the one that maximizes the between-class variance of each component which can be generally defined by

$$\sigma_B^{c\,2} = \sum_{j=1}^{n} w_j^C \left(\mu_j^C - \mu_T^C\right)^2, \tag{6}$$

where j represents a specific class in such a way that w_j^C and μ_j^C are the probability of occurrence and mean of class j, respectively. In other words, the problem of n-level thresholding is reduced to an optimization problem to search for the thresholds t_j^C that maximizes the objective functions (*i.e.*, fitness function) of each image component C, generally defined as

$$\varphi^C = max_{1<t_1^C<\cdots<t_{n-1}^C<L-1}\, {\sigma^{c^2}}_B\left(t_j^C\right). \tag{7}$$

Computing the above optimization problem involves high computational complexity as the number of threshold levels and image components increase. Many optimization methods have been proposed in the literature [2]. However, more recently, biologically inspired methods, such as the well-known *PSO*, have been used as computationally efficient alternatives to analytical methods to solve optimization problems [16, 17].

B. General Approach

The original *PSO* algorithm was developed by Eberhart and Kennedy in 1995 [34]. The *PSO* basically takes advantage of the swarm intelligence concept, which is the property of a system whereby the collective behaviors of unsophisticated agents that are interacting locally with their environment, create coherent global functional patterns. More recently, and based on the concepts inherent to the *PSO*, the *DPSO* [36] and the *FOPSO* [37], an extended version denoted as *FODPSO* was presented in [35], in which several swarms compete using Darwin's survival-of-the-fittest principles and fractional calculus to control the convergence rate of the algorithm. Using those principles, the *FODPSO* enhances the ability of the *PSO* algorithm to escape from local optima by running several simultaneous parallel *PSO* algorithms, each being a different swarm, on the same test problem and apply a simple selection mechanism. When a search tends to a local optimum, the search in that area is simply discarded and another area is searched instead. In this approach, at each step, swarms that show improvement are rewarded (extend particle life or spawn a new descendent) and swarms which stagnate are punished (reduce swarm life or delete particles). Moreover, the approximate *Grünwald–Letnikov FC* definition allows to use the concept of fractional differential with α, $0 \leq \alpha \leq 1$, to control the convergence rate of particles.

Table I presents the *FODPSO* algorithm applied to image segmentation. Each particle, a ,within each different swarm, s, moves in a multidimensional space according to position ($x_a[t]$), $0 \leq x_a[t] \leq L-1$, and velocity ($v_a[t]$). The position and velocity values are highly dependent on the local best ($\breve{x}_a[t]$) and global best ($\breve{g}_a[t]$) information. The coefficients w, ρ_1 and ρ_2 are assigned weights, which control the inertial influence, *i.e.*, according to "the globally best" and "the locally best", respectively, when the new velocity is determined. Typically, the inertial influence is set to a value slightly less than 1. ρ_1 and ρ_2 are constant integer values, which represent "cognitive" and "social" components. However, different results can be obtained by assigning different influences for each component. Depending on the application and the characteristics of the problem, tuning these parameters properly will lead to better results. The parameters r_1 and r_2 are random vectors with each component generally a uniform random number between 0 and 1. The intent is to multiply a new random component per velocity dimension, rather than multiplying the same component with the velocity dimension of each particle.

Table I. The *FODPSO* Segmentation Algorithm.

Initialize α, ρ_1, ρ_2 // fractional coefficient, global and local weights
Initialize N, N_{min}, N_{max} // initial, minimum and maximum number of particles within each swarm
Initialize $N^s, N_{min}^s, N_{max}^s$ // initial, minimum and maximum number of swarm
Initialize Δv // maximum number of levels a particle can travel between iterations
Initialize I_T, I_{kill} // total number of iterations and maximum stagnation of swarms
$p_i^C = \frac{h_i^C}{N},\ \sum_{i=0}^{L-1} p_i^C = 1$
$\mu_T^C = \sum_{i=0}^{L-1} i p_i^C$
Initialize $0 \leq x_a^s[0] \leq L-1$ // initial position of all particles from all swarms
Initialize $\breve{x}_a^s, \breve{g}_a^s$ based on $x_a^s[0]$ // initial local best of all particles and global best of all swarms
For each iteration t until I_T // main loop
  For each particle a of swarm s
    $v_a^s[t+1] = \alpha v_a^s[t] + \frac{1}{2}\alpha v_a^s[t-1] + \frac{1}{6}\alpha(1-\alpha)v_a^s[t-2] + \frac{1}{24}\alpha(1-\alpha)(2-\alpha)v_a^s[t-3] + \rho_1 r_1(\breve{g}_a^s - x_a^s[t]) + \rho_2 r_2(\breve{x}_a^s - x_a^s[t]),\ |v_a^s[t+1]| \leq \Delta v$
    $x_a^s[t+1] = x_a^s[t] + v_a^s[t+1],\ 0 \leq x_a^s[t+1] \leq L-1$
    Compute (4) and (5) based on the thresholds defined in $x_a^s[t+1]$
    ${\sigma^{c^2}}_B = \sum_{j=1}^{n} w_j^C\left(\mu_j^C - \mu_T^C\right)^2$ // compute the solution of each particle a of swarm s
    If ${\sigma^{c^2}}_{a_B} > {\sigma^{c}}_{abest_B}^2$ // particle a has improved
      ${\sigma^{c}}_{nbest_B}^2 = {\sigma^{c^2}}_B$
      $\breve{x}_a^s = x_a^s[t+1]$
  For each swarm s
    If $\max {\sigma_s^{c^2}}_B > \varphi^C$ // swarm s has improved
      $\varphi^C = \max {\sigma_s^{c^2}}_B$
      $\breve{g}_a^s = x_a^s[t+1]$
      $I_k = 0$ // reset stagnancy counter
      If $N_s < N_{max}$ // the current number of particles within swarm s is inferior to the maximum number of allowed particles

$N_s = N_s + 1$
Randomly spawns a new particle in swarm s
If $N^s < N^s_{max}$ and $rand() \frac{N_s}{N_{max}} > rand()$ // small probability of creating a new swarm
$N^s = N^s + 1$
Randomly spawns a new swarm with an initial number of N particles
Else // swarm s has not improved
$I_k = I_k + 1$
If $I_k = I_{kill}$ // swarm s has improved for too long
If $N_s > N_{min}$ // swarm s has currently more than the minimum number of allowed particles to form a swarm
Delete worse particle from swarm s, *i.e.*, lower local solution
Else // swarm s does not currently have the minimum number of allowed particles to form a swarm
Delete whole swarm s, *i.e.*, all particles from swarm s
End

It is noteworthy that the α value greatly affects the inertial particles. With a small α, particles ignore their previous activities, thus ignoring the system dynamics and being susceptible to get stuck in local solutions (*i.e.*, exploitation behavior).). On the other hand, with a large α, particles will present a more diversified behavior, which allows exploration of new solutions and improves the long-term performance (*i.e.*, exploration behavior). However, if the exploration level is too high, then the algorithm may take too much time to find the global solution. Based on the experimental results from [35], it will be used a fractional coefficient of $\alpha = 0.6$, thus resulting in a balance between exploitation and exploration.
One may summarize both computational and memory complexity as Table II depicts.

Table II. Computational and memory complexity of *PSO*, *DPSO* and *FODPSO*.

Complexity	***PSO***	***DPSO***	***FODPSO***
Memory	$\mathcal{O}(C)$	$\mathcal{O}(C)$	$\mathcal{O}(4C)$
Computational	$\mathcal{O}(nN^P)$	$\mathcal{O}(n\sum_{\forall s} N^S)$	$\mathcal{O}(n\sum_{\forall s} N^S)$

Note that the memory complexity of the *FODPSO* is larger than the alternatives since it intrinsically has memory properties related to the fractional extension. In other words, due to the truncation order of the approximate fractional derivative, it needs to track the last 4 steps of each particle's velocity that depends on the number of components C (*i.e.*, bands) of the image. The computational complexity of the algorithms was considered excluding the initial computation of equations (1) and (2). Note that this may be accomplished since the three algorithms require that same initial computation that depends on the size of the image. After that initial setup, the three algorithms may be adjusted in such a way to ensure a similar computational complexity. Likewise, the computational complexity of the three algorithms will increase with the number of desired thresholds n. Nevertheless, while the *PSO* depends on the number of particles N^P within the population, the *DPSO* and *FODPSO* depend on the accumulated number of particles within each swarm, *i.e.*, $\sum_{\forall s} N^S$. In other words, one may ensure that the computational complexity of both *DPSO* and *FODPSO* will be inferior to the *PSO* one by defining the maximum number of particles within each swarm as $N_{max} \leq \frac{N^P}{N^s_{max}}$, wherein N^s_{max} represent the maximum number of allowed swarms. It is, however, noteworthy that one may avoid holding this assumption since the evolutionary features of both *DPSO* and *FODPSO* are stochastic and depends on uniformly distributed variables. In other words, by setting a an adequate combination between the minimum and maximum acceptable number of particles to form a swarm N_{min} and N_{max} and the minimum and maximum number of swarms within the population N^s_{min} and N^s_{max}, one may ensure that $\mathcal{O}(n\sum_{\forall s} N^S) \leq \mathcal{O}(nN^P)$ for a steady state regime. Such a condition may be achieved by adhering to the following condition: $N^P \geq \frac{N^s_{min}N_{min}+N^s_{max}N_{max}}{2}$.

C. Algorithm Evaluation

The computational time is one of the most important indicators along with the fitness value to determine the performance of the algorithm. Provided that the data is large, the efficiency of the method is restricted to a great extent [44]. For instance, hyperspectral images are in general large, so using a high speed and efficient algorithm is highly preferable. Moreover, in real-time applications, using a high-speed algorithm is the main objective [45]. As a result, the evaluation of the *CPU* processing time and the fitness value seems vitally important to show the efficiency of the new method. In addition, since all bio-inspired methods are random and stochastic, the results are not completely the same in each run. Consequently, the stability of different methods should be evaluated by an appropriate index such as standard deviation value.
PSO-based segmentation algorithms have been widely used in recent years. In fact, the ability of the traditional *PSO*-based segmentation has already been compared with other thresholding-based methods such as *GA*-based algorithms and exhaustive ones. Results confirm that the *PSO* based method present better results in terms of fitness value and *CPU* processing time. In [46], authors illustrated that the *PSO*-based segmentation method acted better than other methods such as Genetic Algorithms (*GA*), Differential Evaluation (*DE*), Ant Colony Optimization (*ACO*), Simulated Annealing (*SA*) and Tabu Search (*TS*) in terms of precision, robustness of the results and runtime. In [26], *PSO* outperforms GA in terms of *CPU* time and the fitness value for Kapur's and Otsu's functions. In [47], results indicate that *PSO*-family methods act better than GA with a learning operator (*GA-L*) from different points of view. Consequently, it is easy to detect that *PSO*-based segmentation methods are considered an efficient way to find optimal thresholds in short *CPU* processing time.

III. EXPERIMENTAL RESULTS

To compare the performance of the proposed *FOPSO* method with the *PSO* and *DPSO* approaches, all methods are tested on two different types of images, *i.e.*, multispectral and hyperspectral images. In all cases the image segmentation approaches were programmed in *MATLAB* on a computer having *Intel Core 2 Duo T5800* processor (2.00 *GHz*) and 3*GB* of memory.

A. Description of data sets

a) First Test Case – Multispectral Worldview Image

The first data set is an 8 x 8 km multispectral Worldview satellite image consisting of 8 bands captured at Tamworth, Northern New South Wales in Australia (Fig. 1). The pixel size was 2.4 x 2.4m. The image covered large sections of pine plantations, interspersed with native vegetation, grasslands, logged areas, barren soil and roads. This introduced a high level of natural variability to the segmentation problem. Unlike artificial objects, natural vegetation has multiple levels of variation. For example, within the pine plantation class, there are age differences, differences in reflectance due to slope, aspect, sun position, soil types, etc., and all these cause added complexities in the segmentation scheme. Fig. 1 shows an image of the data where data channels 5, 3 and 2 are used for showing R, G, and B components respectively while Fig. 2 depicts the histogram for all 8 data channels.

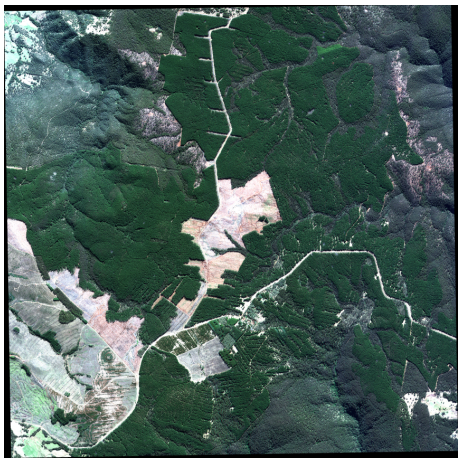

Figure 1: Our test case study site (data channels 5, 3, 2 are mapped to the R, G, and B channels)

Table III gives the initial parameters of the *PSO, DPSO* and the proposed *FODPSO* based methods for the first test case. The *PSO*, *DPSO* and *FODPSO* methods are parameterized algorithms. Therefore, one needs to be able to choose the parameter values that would result in faster convergence. The cognitive, social and inertial weights were chosen by taking into account several works focusing on the convergence analysis of the traditional *PSO* (*cf.*, [35, 47, 1]). For instance, to guarantee the convergence of the process, Jiang *et al.* [47] presented a set of attraction domains that altogether present a relation between $\boldsymbol{\rho_1}$, $\boldsymbol{\rho_2}$ and $\boldsymbol{w}$, wherein $0 \leq \boldsymbol{w} < 1$ and $\boldsymbol{\rho_1} + \boldsymbol{\rho_2} > 0$. Based on the attraction domain in [3], if one would choose an inertial coefficient $\boldsymbol{w} = 0.8$, the sum between the cognitive and social components would need to be less than 7, *i.e.*, $\boldsymbol{\rho_1} + \boldsymbol{\rho_2} < 7$. The parameters in Table II were selected by considering that many works present a larger cognitive coefficient (*cf.*, [47]). Note that the threshold velocities of particles and the maximum number of particles within each swarm in the *DPSO* are smaller than the *PSO* algorithm. This was experimentally adjusted to provide swarms of 20 particles with the same level of diversity (*i.e.*, exploration and exploitation) than swarms of 150 particles.

Table III: Initial parameters of the *PSO, DPDO* and *FODPSO* for the first data set

Parameter	***PSO***	***DPSO***	***FODPSO***
$\boldsymbol{I_T}$	100	100	100
N	150	20	20
$\boldsymbol{\rho_1}$	1.2	1.2	1.2
$\boldsymbol{\rho_2}$	0.8	0.8	0.8
$\boldsymbol{w}$	0.8	-	-
Δv	2	2	2
N_{min}	-	10	10
N_{max}	-	30	30
N^s	-	4	4
N^s_{min}	-	2	2
N^s_{max}	-	6	6
I_{kill}	-	10	10
α	-	-	0.6

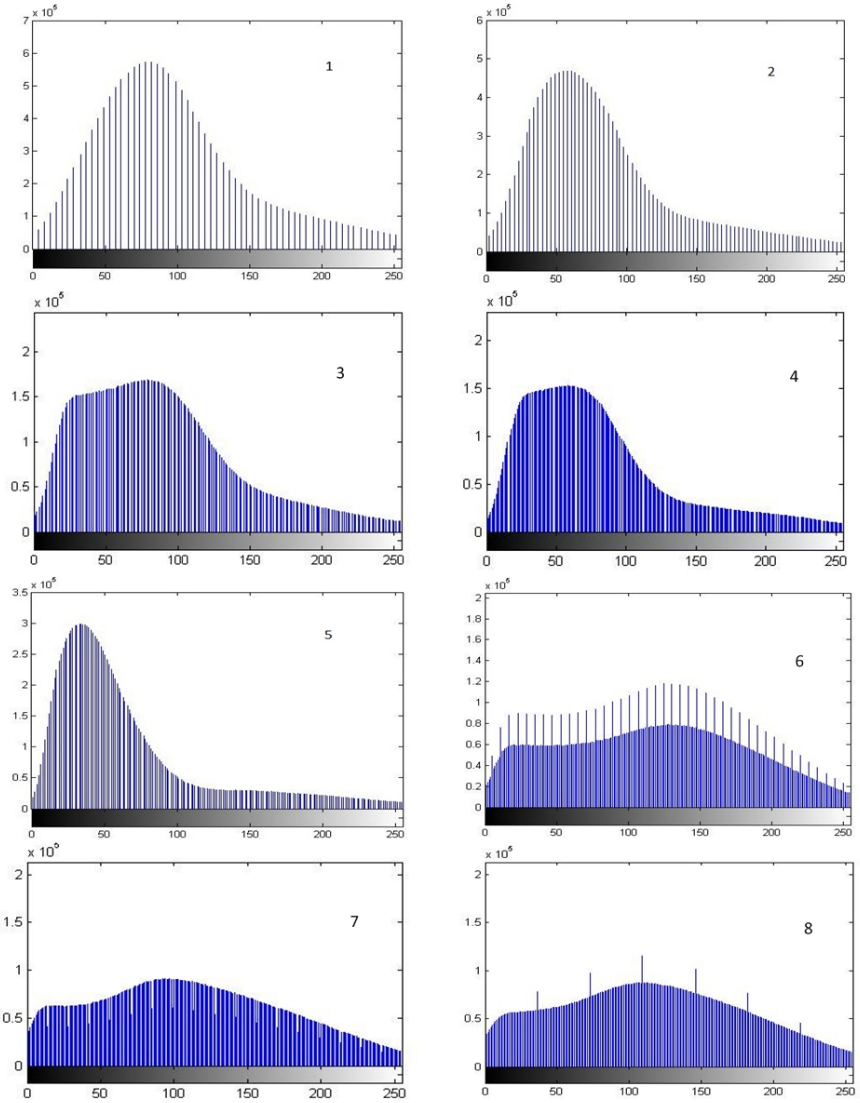

Figure 2: The histograms of different data channels (data channel no. inserted in each figure). Grey values on x-axis and value count on y-axis.

b) Second data set – Hyperspectral ROSIS image

The second test case is a hyperspectral data set which was captured on the city of Pavia, Italy by Airborne data from the ROSIS-03 (Reflective Optics System Imaging Spectrometer). The ROSIS-03 sensor has 115 data channels with a spectral coverage ranging from 0.43 to 0.86 μm. In our experiments we eliminate 12 noisy data channels and used 103 data channels for processing. The spatial resolution is 1.3m per pixel. The original data set is 610 by 340 pixels. This data set is captured on the Engineering School at the University of Pavia consisting of different classes including: trees, asphalt, bitumen, gravel, metal sheet, shadow, bricks, meadow and soil. Fig. 3 shows an image of the second test case.

The proposed multilevel thresholding techniques based on *PSO*, *DPSO and FODPSO* were implemented with the specific parameters shown in Table IV for the second test case. Table III presents the initial parameters of the *PSO*- and *DPSO*-based methods for the second test case. The main differences here in comparison to Table II are that the maximum velocity of particles and the capacity of each swarm, in the case of the *DPSO* and *FODPSO* algorithms, need to be increased in order to overcome the increased complexity of using data.

Fig. 3: an image of the second test case

Table IV: Initial parameters of the *PSO*, *DPSO* and *FODPSO* for the second data set

Parameter	***PSO***	***DPSO***	***FODPSO***
$\boldsymbol{I_T}$	100	100	100
N	150	15	15
$\boldsymbol{\rho_1}$	1.2	1.2	1.2
$\boldsymbol{\rho_2}$	0.8	0.8	0.8
$\boldsymbol{w}$	0.8	-	-
Δv	5	5	5
N_{min}	-	10	10
N_{max}	-	50	50
N^s	-	4	4
N^s_{min}	-	2	2
N^s_{max}	-	6	6
N_{kill}	-	10	10
α	-	-	0.6

B. Results and Discussion

a). First Test Case: Multispectral image

The *CPU* average processing time of the *PSO*, *DPSO* and *FODPSO* for 6-, 8- and 10-level thresholding is presented in Table V and was calculated over 40 different runs. *PSO* is referred to as a fast optimization algorithm. However, as can be seen from Table V, the computation time for *PSO*-based segmentation was significantly higher than for both the *FODPSO* and *DPSO* methods. The main reason for this is that the *PSO* has a fixed population of 150 particles which, in other words, means that 150 different solutions are needed to be evaluated within the same swarm. The *FODPSO* and *DPSO*, on the other hand, are composed of multiple smaller swarms (between 2 and 6 swarms of 10 and 50 particles each), being faster than the *PSO* even with an equal or larger number of particles in the whole *DPSO* and *FODPSO*. The dynamical clustering of particles inherent to both *FODPSO* and *DPSO* allows releasing most of the processing effort necessary to compute the local and global solutions. In other words, the *CPU* processing time decreases as the number of particles within the same swarm decreases. The difference percentages of *CPU* processing time between *FODPSO* and *DPSO* remain almost the same regardless on the segmentation level in the range of 3 to 8 percent. Although the difference may be considered small to justify the choice between the *FODPSO* and the *DPSO*, it still represents an improvement that can be highly pondered depending upon the fitness value of the algorithms. Moreover, the stability of the traditional *PSO* highly deteriorates for a segmentation level of 10, contrarily to both *FODPSO* and *DPSO*.

Table V: Average and STD CPU processing time (in seconds) of each algorithm for different levels

Level	**FODPSO**	**DPSO**	**PSO**	**Difference (%) between FODPSO and DPSO**	**Difference (%) between FODPSO and PSO**
6	41.05 ± 0.63	43.04 ± 0.78	46.69 ± 0.54	4.8	13.73
8	54.15 ± 1.21	56.21 ± 1.27	60.03 ± 0.65	3.8	10.85
10	65.46 ± 0.90	67.12 ± 1.39	73.60 ± 2.22	2.5	12.43

The fitness and optimal threshold values were calculated for all different data channels separately. The average and standard deviation fitness values of all data channels were calculated for each level of segmentation and the obtained results are presented in Table VI. *FODPSO* generally performed slightly better than other methods in terms of fitness value. An exception may be observed for a segmentation level of 6 in which the *DPSO* presented a slightly better result than the *FODPSO*. It is noteworthy that such behavior should be expected for specific situations since the *DPSO* is a particular case of the *FODPSO*. In general, both *DPSO* and *FODPSO* give a better fitness because the *PSO* may get stuck in the vicinities of the global solution, while both *FODPSO* and *DPSO* use natural selection in order to avoid stagnation (*cf.*, [35, 36]). Hence, it can be concluded that both Darwinian algorithms are able to find better thresholds in less *CPU* time than the traditional *PSO*. Fig. 4 shows 10-level segmented image based on *FODPSO* and their histograms.

Table VI: Average fitness values for all bands at each level for the first test case

Level	FODPSO	DPSO	PSO
6	3812.23 ± 0.25	3812.31 ± 0.04	3811.45 ± 0.59
8	3877.56 ± 0.23	3877.21 ± 0.07	3876.04 ± 0.60
10	3907.52 ± 0.24	3907.15 ± 0.15	3905.95 ± 0.61

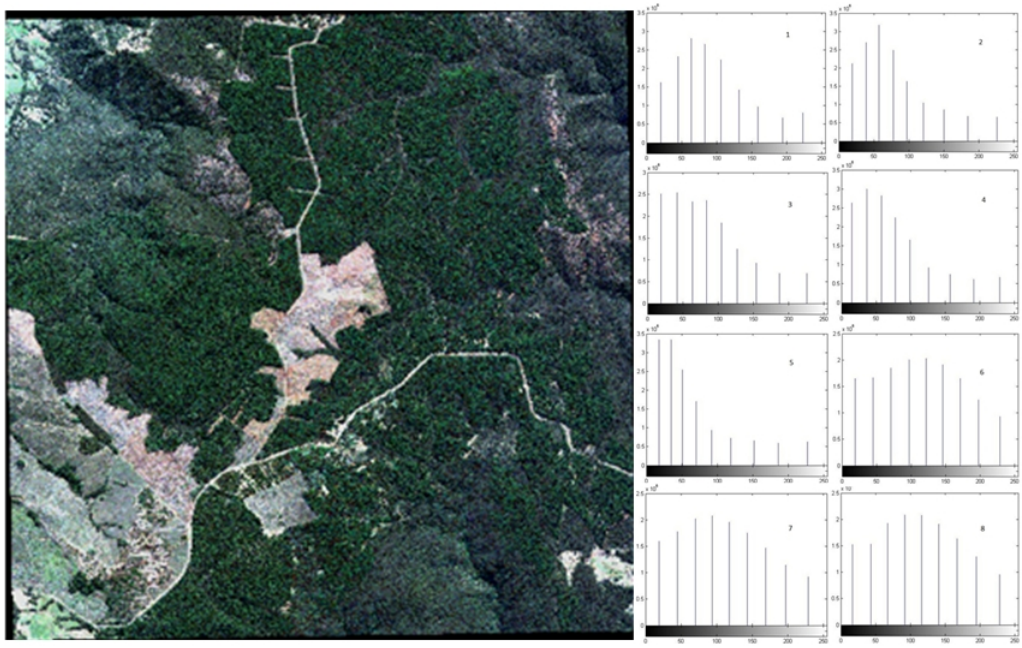

Figure 4**:** 10-level segmented image (data channels 5, 3, 2 are mapped to the R, G, and B channels of the display) based on *FODPSO* and the histograms of all data channels

Despite the minor differences in fitness values between the *FOPSO* and the *DPSO* with respect to the between-class variance, one should note that the *FODPSO*-based thresholding is able to achieve segmentation of the image faster than both *DPSO* and *PSO*. Consequently, the proposed *FODPSO* method is very attractive for image segmentation, especially for more complex images and/or high segmentation levels.

Fig. 5 shows a subset of the main image, 6-level and 10-level *FODPSO* based segmented images zoomed by 200 percent. As can be seen from the figure, the main image (Fig. 5c) has more details than the other images. In contrast, the 6 level segmented image (Fig. 5a) is the roughest image. It is easy to conclude that by increasing the level of segmentation, the segmented image includes more detail. As a result, the 10-level segmented image (Fig. 5b) is smoother than the 6 level one. Our segmented image is also less pixelated compared to the original image.

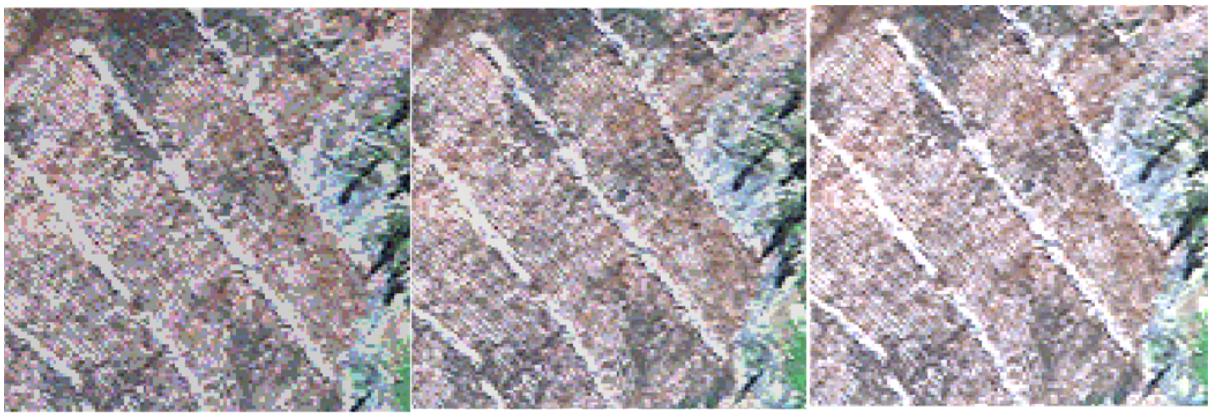

Figure 5: The subset of a) 6-level, b) 10-level c) input image zoomed by200 percent

To further improve the comparison between the three algorithms, the significance of the segmentation method and the segmentation level (independent variables) on the fitness value and the *CPU* processing time (dependent variables) was analyzed using the two-way *MANOVA* technique after checking the assumptions of multivariate normality and homogeneity of variance/covariance [49, 50]. The assumption of normality for each of the univariate dependent variables was examined using univariate tests of *Kolmogorov-Smirnov* (*p-value* < 0.05). The univariate normality of each dependent variable has not been verified. However, since $n \geq 30$ the multivariate normality was assumed based on the Central Limit Theorem (*CLT*) [49-51]. Furthermore, the assumption of multivariate normality was validated [49, 50]. The assumption about the homogeneity of variance/covariance matrix in each group was examined with the *Box's M Test* (M = 605.13, F(24; 376576.64) = 24.693; p-value = 0.001). Although the homogeneity of variance/covariance matrices has not been verified (*i.e.*, *p*-value = 0.001), the *MANOVA* technique is robust to this violation because all the samples have the same size [49, 50]. When the *MANOVA* detected significant statistical differences, we proceeded to the commonly-used *ANOVA* for each dependent variable followed by the *Tukey's HSD Post Hoc*. The classification of the size effect (*i.e.*, measure of the proportion of the total variation in the dependent variable explained by the independent variable) was done according to Maroco [49] and Pallant [50]. This analysis was performed using the *IBM SPSS Statistics* software with a significance level of 5%.

A two-way *MANOVA* analysis was carried out to assess whether the algorithms used on this study have statistically significant differences with respect to the segmentation process. The *MANOVA* analysis revealed that the type of algorithm had a large effect and significant on the multivariate composite (*Pillai's Trace* = 0.973; F(4; 702) = 166.19; *p-value* = 0.001; *Partial Eta Squared* η_p^2 = 0.486; *Power* = 1.0). The segmentation level had a very large and significant effect on the multivariate composite (*Pillai's Trace* = 1.847; F(4; 702) = 2116.515; *p-value* = 0.001; η_p^2 = 0.923; *Power* = 1.0). Finally, the interaction between the two independent variables had a moderate effect and significant on the multivariate composite (*Pillai's Trace* = 0.469; F(8; 702) =26.901; *p-value* = 0.001; η_p^2 = 0.235; *Power* = 1.0).

After observing the multivariate significance for different algorithm types and segmentation levels, a univariate *ANOVA* for each dependent variable followed by the *Tukey's HSD Test* was carried out. For the type of algorithm, the dependent variable fitness value presents statistically significant differences (F(2, 351) = 469.97; *p-value* = 0.001; η_p^2 = 0.728; *Power* = 1.0), as well as the dependent variable *CPU* processing time (F(2, 351) = 2138.04; *p-value* = 0.001; η_p^2 = 0.92; *Power* = 1.0). For the segmentation level, the dependent variable fitness value also demonstrates statistically significant differences (F(2, 351) = 2445064.03; *p-value* = 0.001; η_p^2 = 1; *Power* = 1.0), as well as the dependent variable *CPU* processing time (F(2, 351) = 1864.22; *p-value* = 0.001; η_p^2 = 0.99, *Power* = 1.0). Using the Tukey's *HSD* Post Hoc, it is possible to verify the differences between the algorithms. Analyzing the fitness value and the *CPU* processing time, there are statistically significant differences between the obtained experimental results using the *PSO*, *DPSO* and *FODPSO* segmentation algorithms.

Table VII: Tukey's HSD Post Hoc Test to the Maximum Communication Distance

Algorithm	Fitness Value	CPU Time
PSO *vs* DPSO	-1.06*	4.61*
PSO *vs* FODPSO	-1.25*	6.34*
DPSO *vs* FODPSO	-0.18*	1.73*

* The corresponding mean difference is significant at the 0.05 level

All *p*-values corresponding to the mean differences are equal to 0.001

It is noteworthy that the *FODPSO* produces better solutions than both the *PSO* and *DPSO*. As expected, the *FODPSO* algorithm pro-

duces better solutions than the *DPSO* and, on the other hand, this last one produces better solutions than the *PSO*. In fact, using the *PSO* segmentation algorithm proves to be the "worse" segmentation method.

As shown in Table VII, which is based on *Tukey's HSD Post Hoc* test, the *FODPSO* is able to reach a slightly better fitness solution in less time. Nevertheless, the differences between the algorithms are not clearly seen in Fig. 6. Although it is possible to observe significant differences in the global *CPU* processing time between the *FODPSO* and the other algorithms, the improvement of the solution is not perceptible. Hence, in the next section the same analysis will be formed on a hyperspectral image.

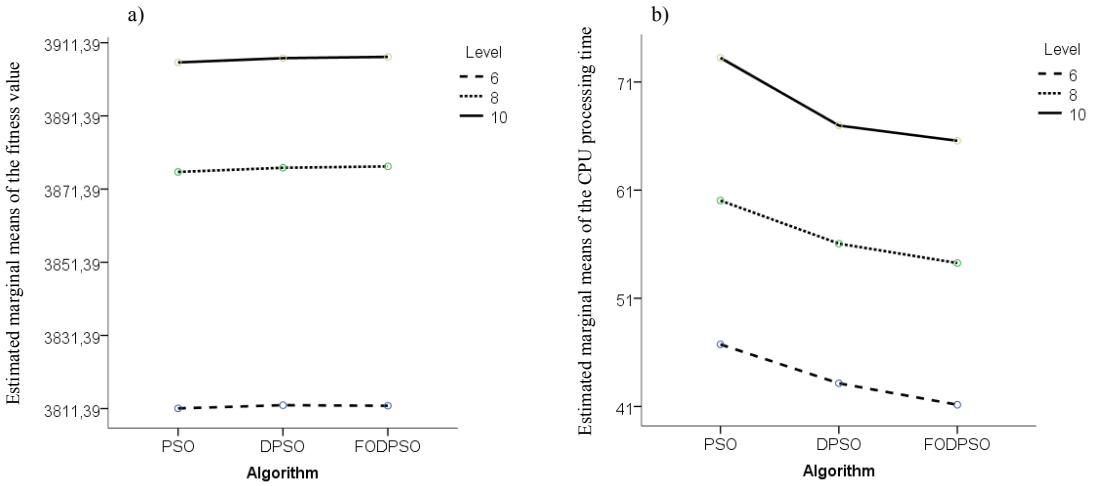

Figure 6: Estimated marginal means of the: a) fitness value; b) CPU processing time.

b) Second data set: Hyperspectral Image

As for the first data set, the *CPU* processing time in the second test case for each algorithm for 10-, 12- and 14-level thresholding was calculated as the average value of 40 different runs and the results being presented in Table VIII. According to Table VIII, the *FODPSO* based method has the least *CPU* processing time in comparison with other studied methods as was observed for the first data. On the contrary, *PSO* is the worst method among others in terms of *CPU* processing time. As can be seen from Table VIII, *FODPSO* significantly outperforms the *PSO* based method in particular when the level of segmentation increases. *FODPSO* improves the result of the *PSO* based segmentation method by 119.6% and 65.1% in the best and worst case respectively. In the same way, the *CPU* processing time of the *FODPSO* is considerably less than for the *DPSO* and shows an improvement by 7.4% and 31.5% for the best and worst case, respectively.

Table VIII: Average and STD CPU processing time for each algorithm and different levels

Level	FODPSO	DPSO	PSO	Percentage Difference between FODPSO and DPSO	Percentage Difference between FODPSO and PSO
10	689.83 ± 66.22	740.62 ± 49.73	1138.81 ± 103.02	7.4	65.1
12	691.96 ± 7.6	800.84 ± 8.4	1387.84 ± 123.43	15.7	100.1
14	753.59 ± 37.51	991.02 ± 82.60	1654.89 ± 141.82	31.5	119.6

Table IX gives information regarding the average fitness value of 103 data channels in 40 different iterations. As in the case of the first multispectral data set, in the hyperspectral test case, *FODPSO* finds optimal threshold values which are better than for the other methods. This shows that *FODPSO* is able to find optimal thresholds with better fitness values in less *CPU* processing time compared to the other studied methods. The fitness value of the *FODPSO* based method is followed by *DPSO* which is more efficient than the conventional *PSO*. As can be seen from the table, by increasing the level of segmentation, the fitness of *FODPSO* increases more than the fitness of the other methods. *PSO* gives almost the same fitness for 10-, 12- and 14- level of segmentation since it is not endowed with any kind of mechanism to improve the convergence of particles when in the vicinities of the optimal solution.

Table IX: Average and STD fitness values at each level

Level	FODPSO	DPSO	PSO
10	2971.69 ± 1.13	2971.22 ± 0.40	2970.09 ± 0.12
12	3002.73 ± 5.14	2991.78 ± 0.65	2984.92 ± 0.90
14	3090.13 ± 14,06	3035.97 ± 10.65	2997.53 ± 3.04

Fig. 7 shows 10-level and 14-level *FODPSO* based segmented images using a 200 percent zoom. As can be seen from the figure, 14-

level based segmented image (Fig. 7(b)) provides more details than the 10-level segmentation.

(a) (b)

Figure 7: The subset of a) 10-level, b) 14-level FODPSO based segmented image zoomed by200 percent

Similarly to the first data set, the assumption of normality for each of the univariate dependent variables was examined using univariate tests of *Kolmogorov-Smirnov* (*p-value* < 0.05) [49-51]. The assumption about the homogeneity of variance/covariance matrix in each group was examined with the *Box's M Test* (M = 1239.38, F(24; 376576.64) = 50.58; p-value = 0.001). When the *MANOVA* detected significant statistical differences, we proceeded to the commonly-used *ANOVA* for each dependent variable followed by the *Tukey's HSD Post Hoc*.

The *MANOVA* analysis revealed that the algorithm type had a very large and significant effect on the multivariate composite (*Pillai's Trace* = 1.40; F(4; 702) = 405.97; *p-value* = 0.001; *Partial Eta Squared* η_p^2 = 0.698; *Power* = 1.0). The segmentation level also had a large and significant effect on the multivariate composite (*Pillai's Trace* = 0.97; F(4; 702) = 165.03; *p-value* = 0.001; η_p^2 = 0.49; *Power* = 1.0). Finally, the interaction between the two independent variables had a very large and significant effect on the multivariate composite (*Pillai's Trace* = 1.02; F(8; 702) = 91.82; *p-value* = 0.001; η_p^2 = 0.51; *Power* = 1.0).

After observing the multivariate significance in the type of algorithm and the segmentation level, a univariate *ANOVA* for each dependent variable followed by the *Tukey's HSD Test* was carried out. For the type of algorithm, the dependent variable fitness value presents statistically significant differences (F(2, 351) = 1066.64; *p-value* = 0.001; η_p^2 = 0.86; *Power* = 1.0), as well as the dependent variable *CPU* processing time (F(2, 351) = 2309.24; *p-value* = 0.001; η_p^2 = 0.93; *Power* = 1.0). For the segmentation level, the dependent variable fitness value also presents statistically significant differences (F(2, 351) = 3907.10; *p-value* = 0.001; η_p^2 = 0.96; *Power* = 1.0), as well as the dependent variable CPU processing time (F(2, 351) = 77.58; *p-value* = 0.001; η_p^2 = 0.66, *Power* = 1.0).

Using the Tukey's *HSD* Post Hoc, one can observe that there are statistically significant differences between experiments using the *PSO*, *DPSO* and *FODPSO* segmentation algorithms, for both CPU processing time and fitness function.

Table X: Tukey's HSD Post Hoc Test to the Maximum Communication Distance

Algorithm	Fitness Value	CPU Time
PSO *vs* DPSO	-15.47*	549.69*
PSO *vs* FODPSO	-37.33*	682.05*
DPSO *vs* FODPSO	-21.86*	132.37*

* The corresponding mean difference is significant at the 0.05 level

All *p*-values corresponding to the mean differences are equal to 0.001

Once again, the *FODPSO* produces better solutions than both the *PSO* and *DPSO in terms of fitness value*. Furthermore, as expected, the DPSO produces better solutions than the *PSO*. As shown in Table X (also shown in Fig. 8) based on *Tukey's HSD Post Hoc* test, the fractional-order algorithm is able to once again reach a better fitness solution in less time. Moreover, the differences between the *FODPSO* and the other algorithms are more evident as the segmentation level increases. This should be highly appreciated as many applications require real-time multi-segmentation methods (*e.g.*, autonomous deployment of sensor nodes in a given environment).

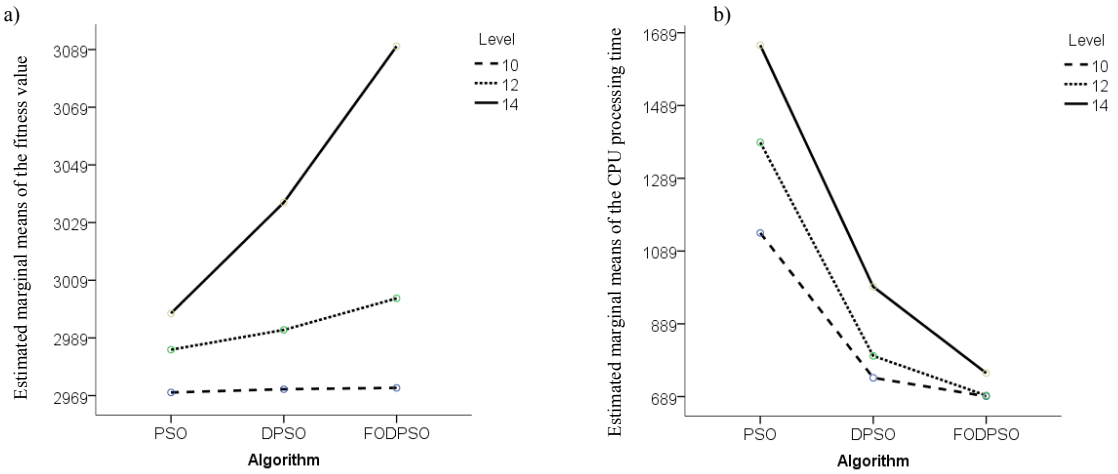

Figure 8: Estimated marginal means of the: a) fitness value; b) CPU processing time.

In summary, it is possible to observe that the *FODPSO* is faster than the *DPSO* since fractional calculus is used to control the convergence rate of the algorithm. As described in [48], a swarm behavior can be divided into *exploitation* and *exploration*. The exploitation behavior is related with the convergence of the algorithm, allowing a good short-term performance. However, if the exploitation level is too high, then the algorithm may be stuck on local solutions. On the other hand, the exploration behavior is related with the diversification of the algorithm which allows exploring new solutions, thus improving the long-term performance. However, if the exploration level is too high, then the algorithm may take too much time to find the global solution. In the *DPSO*, the trade-off between exploitation and exploration can only be handled by adjusting the inertia weight w. While a large inertia weight improves exploration activity, the exploitation may be improved using a small inertia weight. Since the *FODPSO* presents a fractional calculus strategy to control the convergence of particles with memory effect, the coefficient α allows providing a higher level of exploration while ensuring the global solution of the algorithm (*cf.*, [36]).

IV. CLASSIFICATION

Although the main idea behind this work is to introduce a thresholding base segmentation technique, it is of interest to see the effectiveness of the new segmentation method on classification. In this way, this section presents a novel framework to prove the efficiency of the proposed method for classification. The proposed classification method is based on the *FODPSO* and the Support Vector Machine (SVM) classifier. Since we do not have reference samples for the first data set, the classification is only performed on the second data set. Fig. 9 shows the general idea of the proposed classification approach. As can be seen, the data have been first classified with *SVM* and a Gaussian kernel. The hyper parameters have been selected using 5-fold cross validation. Each variable has been scaled between -1 and 1. To carry out a fair evaluation, the input is classified only once while the output of this step is used for all different levels. By doing that, the accuracy of the classification for different methods is only dependent on the effect of the segmentation method. In parallel, the input data is transformed using the Principal Component Analysis *(PCA)* and the first Principal Component *(PC)* is kept since the most of the variance is provided by that. The output of this step is segmented by proposed *FODPSO* method. In the final step, the results of the *SVM* and the *FODPSO* are combined by using Majority Voting (*MV*).

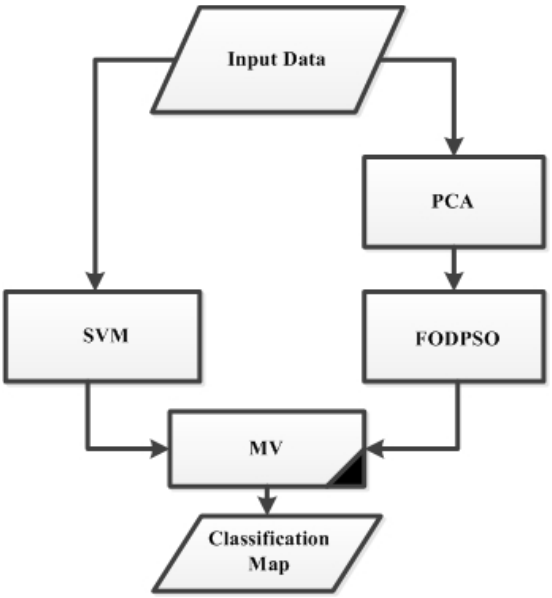

Fig. 9: Illustrative flowchart of the new classification approach.

Fig. 10 depicts the general idea of the proposed approach with *MV*. The output of segmentation methods is a few number of objects and each object consist of several pixels with the same label. In other words, pixels in each object share the same characteristics. To perform the *MV* on the output of the segmentation and classification steps, a counting on the number of pixels with different class labels in each object is first carried out. Subsequently, the all pixels in each object are assigned to the most frequent class label for the object. In the case where two classes have the same (most frequent) proportions in one object, the object is not assigned to any of those classes and the result of the traditional SVM is considered for each pixel in the object directly.

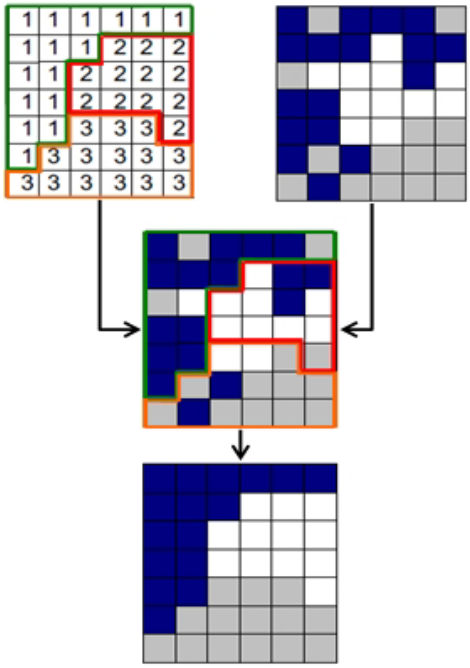

Fig. 10: Integration of the classification and segmentation steps using *MV* [4].

The procedure of the new classification approach is described step-by-step as it follows:

I. The input data is classified by *SVM*;
II. The input data is transformed by *PCA* and the first *PC* is kept;
III. The output of *II* is segmented by *FODPSO*;
IV. The result of *I* and *III* are combined using *MV*;

Fig. 11 illustrates the classification map of the standard *SVM* and the proposed classification method with 10-, 12- and 14-level segmentation by *FODPSO*. The output of the *SVM* presents a lot of noisy pixels which decreases the accuracy of the classification. The result of the overall accuracy and kappa coefficient for the *SVM* and the new method with 10-, 12-, and 14-levels are shown in Table X. For a better understanding, the classification accuracy for each class is also included in the table. All 3 segmentation levels improve the result of the *SVM* classification. The accuracy increases when the number of levels increases from 10 to 14. The main reason behind that phenomenon is denoted as *under segmentation* in which several objects are merged into a single one. This problem can be easily solved by increasing the number of levels. *SVM+FODPSO* with 10-, 12-, and 14-levels improve the kappa coefficient of *SVM* by 1, 2.2 and 2.6, respectively.

Fig. 11: The classification map of the standard SVM and the proposed classification method with 10-, 12- and 14-level segmentation by FODPSO

Table X: The result of the standard SVM and the proposed classification method with 10-, 12- and 14-level of segmentation by FODPSO. Classification accuracies are given in percentage.

Class		Samples		SVM	SVM+ FODPSO(10)	SVM+ FODPSO(12)	SVM+ FODPSO(14)
No	**Name**	**Training**	**Test**				
1	**Asphalt**	840	5791	94.4	87.1	95.4	96.1
2	**Meadow**	2317	16332	98.1	96.9	97.6	96.9
3	**Gravel**	253	1846	77.9	98.8	98.2	98.5
4	**Trees**	373	2691	93.0	99.7	98.6	99.0
5	**Metal sheets**	149	1196	99.2	100	99.9	100
6	**Bare soil**	619	4410	89.4	96.0	93.7	97.4
7	**Bitumen**	181	1149	85.8	97.3	97.9	97.8
8	**Bricks**	480	3202	92.0	91.0	87.4	86.5
9	**Shadow**	135	812	99.4	99.1	99.9	99.9
Overall Accuracy				*94.3*	95.0	*96.0*	96.2
Kappa Coefficient				*92.4*	93.4	*94.68*	95.0

V. CONCLUSION

In this paper, a novel multilevel thresholding segmentation method is proposed for grouping the pixels of multispectral and hyperspectral images into different homogenous regions. The new method is based on *Fractional-Order Darwinian Particle Swarm Optimization* (*FODPSO*) which is used for finding the optimal set of threshold values and uses many swarms of test solutions which may exist at any time. In the *FODPSO*, each swarm individually performs just like an ordinary Particle Swarm Optimization (*PSO*) algorithm with a set of rules governing the collection of swarms that are designed to simulate natural selection. Moreover, the concept of fractional derivative is used to control the convergence rate of particles. Experimental results compare the *FODPSO* with the classical *PSO* and Darwinian *PSO* (*DPSO*) within multi-level segmentation problems on remote sensing images from different points of view such as *CPU* time and corresponding fitness value. Segmentation methods were carried out on two different test cases. The first test case was a multispectral image related to native vegetation, grasslands, logged areas and barren soil. The second test case was a hyper-

spectral image which is from an urban area, showing a wide variety of human artifacts. Experimental results indicate that the *FODPSO* is more robust than the two other methods and has a higher potential for finding the optimal set of thresholds with more between-class variance in less computational time, especially for higher segmentation levels and for images with a wide variety of intensities. In addition, to show the efficiency of the proposed segmentation method on the result of classification, a novel classification approach based on the new segmentation method and Support Vector Machines (*SVM*) is proposed. Results confirm that the new segmentation method improves the *SVM* in terms of classification accuracies when compared to the standard SVM classification of the raw image data. It should be noted that this is the first time that the concept of *FODPSO* is used in remote sensing, thus showing the potential of its use in efficient image segmentation to determine broad groups of objects. As future work, due to the low computational complexity of the algorithm, the *FODPSO* will be evaluated in image segmentation applications for the real-time autonomous deployment and distributed localization of sensor nodes. The objective is to deploy the nodes only in the terrains of interest, which are identified by segmenting the images captured by a camera on board of an unmanned aerial vehicle using the *FODPSO* algorithm. Such a deployment has importance for emergency applications, such as disaster monitoring and battlefield surveillance. In addition, finding a way for the estimation of the number of thresholds (parameter *n*) and joint multichannel segmentation instead of segmenting data set band by band would be of interest.

Acknowledgment

The Worldview and ROSIS data and corresponding reference information were obtained from Dr. Lalit Kumar of the University of New England, Australia and Prof. Paolo Gamba of the University of Pavia, Italy, respectively. This work was supported in part by the Icelandic Research Fund for Graduate Students, PhD scholarship, (SFRH/BD /73382/2010) from the Portuguese Foundation for Science and Technology (FCT), the Institute of Systems and Robotics (ISR), the Institute of Telecommunications (IT-Covilhã) and RoboCorp.

REFERENCES

[1] P. Ghamisi, M.S. Couceiro, J.A. Benediktsson, and N.M.F. Ferreira, "An Efficient Method for Segmentation of Images Based on Fractional Calculus and Natural Selection", *Expert Systems with Application* Publisher: Elsevier, vol. 39, no. 16, pp. 12407-12417, 2012.

[2] M. Sezgin, and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *J. Electron. Imag*., vol. 13, no. 1, pp. 146–168, Jan. 2004.

[3] Y. Tarabalka, J. Chanussot, and J.A. Benediktsson, "Segmentation and Classification of Hyperspectral Images Using Watershed Transformation," *Pattern Recognition*, vol. 43, pp. 2367-2379, 2010.

[4] M. Fauvel, Y. Tarabalka, J.A. Benediktsson, J. Chanussot, and J.C. Tilton, "Advances in Spectral-Spatial Classification of Hyperspectral Images," in *Proceedings of the IEEE*, vol. 101, pp. 652-675, 2013.

[5] A. Darwish, K. Leukert, and W. Reinhardt, "Image segmentation for the purpose of object-based classification," in *Proceedings of IGARSS'03*, vol. 3, pp. 2039–2041, 2003.

[6] J. Tilton, "Analysis of hierarchically related image segmentations," in *IEEE Workshop on Advances in Techniques for Analysis of Remotely Sensed Data*, pp. 60–69, 2003.

[7] M. Pesaresi and J.A. Benediktsson, "A new approach for the morphological segmentation of high-resolution satellite imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 39, no. 2, pp. 309–320, 2001.

[8] G. Noyel, J. Angulo, and D. Jeulin, "Morphological segmentation of hyperspectral images," *Image Anal. Stereol*, vol. 26, pp. 101–109, 2007.

[9] H. G. Akcay, and S. Aksoy, "Automatic detection of geospatial objects using multiple hierarchical segmentations," *IEEE Trans. Geosci. Remote Sens*, vol. 46, no. 7, pp. 2097–2111, 2008.

[10] J. Chanussot, and P. Lambert, "Bit mixing paradigm for multivalued morphological filters," in *Proceedings of IEEIPA'97*, pp.804–808, 1997.

[11] J. Chanussot, and P. Lambert, "Total ordering based on space filling curves for multivalued morphology," in *Proceedings of ISMM'98*, pp. 51–58, 1998.

[12] P. Lambert, and J. Chanussot, "Extending mathematical morphology to color image processing," in *Proceedings of CGIP'00*, pp. 158–163, 2000.

[13] A.G. Hanbury, and J. Serra, "Morphological operators on the unit circle," *IEEE Trans. Image Process*, vol. 10, no. 12, pp. 1842–1850, 2001.

[14] J. Angulo, and J. Serra, "Morphological coding of color images by vector connected filters," in *Proceedings of ISSPA'2003*, vol. 1, pp. 69–72, 2003.

[15] E. Aptoula, and S. Lefevre, "A comparative study on multivariate mathematical morphology," *Pattern Recognition*, vol. 40, no. 11, pp. 2914–2929, 2007.

[16] R. V. Kulkarni, and G. K. Venayagamoorthy, "Bio-Inspired Algorithms for Autonomous Deployment and Localization of Sensor," *IEEE trans. On systems*, vol. 40, no. 6, pp. 663-675, 2010.

[17] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong, "A new method for gray-level picture thresholding using the entropy of the histogram," *Computer Vision Graphics Image Processing*, vol. 2, pp. 273–285, 1985

[18] J. Kittler, and J. Illingworth, "Minimum error thresholding," *Pattern Recognition*, vol. 19, pp. 41–47, 1986.

[19] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, Cybernet,* SMC, vol. 9, pp. 62–66, 1979.

[20] T. Pun, "A new method for grey-level picture thresholding using the entropy of the histogram," *Signal Processing*, vol. 2, pp. 223–237, 1980.

[21] T. Pun, "Entropy thresholding: A new approach," *Computer Vision Graphics Image Processing*, vol. 16, pp. 210–239, 1981.

[22] Y. K. Lim, and S. U. Lee, "On the color image segmentation algorithm based on the thresholding and the fuzzy c-means techniques," *Pattern Recognition*, vol. 23, pp. 935–952, 1990.

[23] D. M. Tsai, "A fast thresholding selection procedure for multimodal and unimodal histograms," *Pattern Recognition Letters*, vol. 16, pp. 653–666, 1995.

[24] P. Y. Yin, and L. H. Chen, "New method for multilevel thresholding using the symmetry and duality of the histogram," *Journal of Electronics and Imaging*, vol. 2, pp. 337–344, 1993.

[25] A. D. Brink, "Minimum spatial entropy threshold selection," in *IEEE Proceedings on Vision Image and Signal Processing*, vol. 142 (1995), 128-132, 1995.

[26] Q. Hu, Z. Hou, and W. Nowinski, "Supervised range-constrained thresholding," *IEEE Transactions on Image Processing*, vol. 15, pp. 228–240, 2006.

[27] P. K. Saha, and J. K. Udupa, "Optimum image thresholding via class uncertainty and region homogeneity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 689–706, 2001.

[28] O. J. Tobias, and R. Seara, "Image segmentation by histogram thresholding using fuzzy sets," *IEEE Transactions on Image Processing*, vol. 11, pp. 1457–1465, 2002.

[29] D. B. Fogel, "Evolutionary computation: Toward a new philosophy of machine intelligence," *Second edition, Piscataway*, *NJ: IEEE Press*, 2000.

[30] C. C. Lai, and D. C. Tseng, "A hybrid approach using Gaussian smoothing and genetic algorithm for multilevel thresholding," *International Journal of Hybrid Intelligent Systems* 1, vol. 3 (2004), pp. 143-152, 2004.

[31] P. Y. Yin, "A fast scheme for optimal thresholding using genetic algorithms," *Signal processing*, vol. 72 (1999), pp. 85−95, 1999.
[32] Y. Kao, E. Zahara and I. Kao, "A hybridized approach to data clustering," *Expert Systems with Applications*, vol. 34 (2008), pp. 1754-1762, 2008.
[33] D. Floreano, and C. Mattiussi, "Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies," *Cambridge, MA: MIT Press*, 2008.
[34] J. Kennedy, and R. Eberhart, "A new optimizer using particle swarm theory," in *Proceedings of the IEEE Sixth International Symposium on Micro Machine and Human Science*, pp. 39-43, 1995.
[35] M. S. Couceiro, N. M. F. Ferreira, and J. A. T. Machado. "Fractional Order Darwinian Particle Swarm Optimization," in *Symposium on Fractional Signals and Systems* (FSS'11), November 4-5, Coimbra, Portugal, 2011.
[36] J. Tillett, T. M. Rao, F. Sahin, R. Rao, and S. Brockport, "Darwinian Particle Swarm Optimization," in *Proceedings of the 2nd Indian International Conference on Artificial Intelligence*, pp. 1474-1487, 2005.
[37] E.J.S. Pires, J.A.T. Machado, P.B.M. Oliveira, J.B. Cunha, L. Mendes, "Particle swarm optimization with fractional-order velocity," *Journal on Nonlinear Dynamics*, vol. 61, pp. 295–301,2010.
[38] J. Sabatier, O. P. Agrawal, and J.A.T. Machado, (eds.). "Advances in Fractional Calculus - Theoretical Developments and Applications in Physics and Engineering," *Springer*, 2007.
[39] M. D. Ortigueira, and J.A.T. Machado, "Special Issue on Fractional Signal Processing," *Signal Process*, vol. 83, pp. 2285- 2480, 2003.
[40] J.A.T. Machado, M.F. Silva, R.S. Barbosa, I.S. Jesus, C.M. Reis, M.G. Marcos, and A.F. Galhano, "Some Applications of Fractional Calculus in Engineering," *Hindawi Publishing Corporation Mathematical Problems in Engineering*, pp. 1-34, 2010.
[41] I. Podlubny, "Fractional Differential Equations," *Mathematics in Science and Engineering*, vol. 198, San Diego, California, Academic Press, 1999.
[42] L.Debnath, "Recents Applications of Fractional Calculus to Science and Engineering," *Int. J. Math. Sci.*, vol. 54, pp. 3413- 3442, 2003.
[43] M.S. Couceiro, N.M.F. Ferreira, and J.A.T. Machado, "Aplication of Fractional Algoritms in the Control of a Robotic Bird," *Journal of Communications in Nonlinear Science and Numerical Simulation-Special Issue*, vol. 15, no. 4, pp. 895-910, 2010.
[44] J. Fan, M. Han, and J. Wang, "Single point iterative weighted fuzzy C-means clustering algorithm for remote sensing image segmentation," *Pattern Recognition*, vol. 42 (2009), pp. 2527 – 2540, 2009.
[45] R. V. Kulkarni, and G. K. Venayagamoorthy, "Bio-inspired Algorithms for Autonomous Deployment and Localization of Sensor Nodes," *IEEE Trans. SMC-C*, vol. 40, no. 6, pp. 663-675, November, 2010.
[46] K. Hammouche, M. Diaf, and P. Siarry, "A comparative study of various meta-heuristic techniques applied to the multilevel thresholding problem," *Engineering Applications of Artificial Intelligence*, vol. 23 (2010), pp. 676–688, 2010.
[47] M. Jiang, Y.P. Luo, and S.Y. Yang, "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm," *Information Processing Letters*, vol. 102, no. 1, pp. 8-16, 2007.
[48] K. Yasuda, N. Iwasaki, G. Ueno, and E. Aiyoshi, "Particle Swarm Optimization: A Numerical Stability Analysis and Parameter Adjustment Based on Swarm Activity," *IEEJ Transactions on Electrical and Electronic Engineering*, Wiley InterScience, vol. 3, pp. 642-659, 2008.
[49] J. Maroco, "Análise Estatística com utilização do SPSS," Lisboa: *Edições Silabo*, 2010.
[50] J. Pallant, "SPSS Survival Manual," *Kindle Edition ed. Open University Press*, 4 edition, 2011.
[51] A. C. Pedrosa, and S. M. A. Gama, "Introdução Computacional à Probabilidade e Estatística," *Portugal: Porto Editora*, 2004.