Particle Swarm Optimization and Differential Evolution for model-based object detection

Roberto Ugolotti, Youssef S.G. Nashed, Pablo Mesejo, Spela Ivekovič, Luca Mussi, Stefano Cagnoni

Particle Swarm Optimization and Differential Evolution for Model-based Object Detection

Roberto Ugolotti[a,*], Youssef S. G. Nashed[a], Pablo Mesejo[a], Špela Ivekovič[c], Luca Mussi[a,b], Stefano Cagnoni[a]

[a] *Department of Information Engineering, University of Parma, Viale G.P. Usberti 181a, 43124, Parma, Italy*
[b] *Henesis s.r.l., Viale dei Mille 108, 43125, Parma, Italy*
[c] *Department of Mechanical & Aerospace Engineering, University of Strathclyde, Glasgow G1 1XJ, UK*

Abstract

Automatically detecting objects in images or video sequences is one of the most relevant and frequently tackled tasks in computer vision and pattern recognition.

The starting point for this work is a very general model-based approach to object detection. The problem is turned into a global continuous optimization one: given a parametric model of the object to be detected within an image, a function is maximized, which represents the similarity between the model and a region of the image under investigation.

In particular, in this work, the optimization problem is tackled using Particle Swarm Optimization (PSO) and Differential Evolution (DE). We compare the performances of these optimization techniques on two real-world paradigmatic problems, onto which many other real-world object detection problems can be mapped: hippocampus localization in histological images and human body pose estimation in video sequences. In the former, a 2D deformable model of a section of the hippocampus is fit to the corresponding region of a histological image, to accurately localize such a structure and analyze gene expression in specific sub-regions. In the latter, an articulated 3D model of a human body is matched against a set of images of a human performing some action, taken from different perspectives, to estimate the subject's posture in space.

Given the significant computational burden imposed by this approach, we implemented PSO and DE as parallel algorithms within the nVIDIA^{TM}CUDA computing architecture.

Keywords: Object Detection, Pose Estimation, Deformable Models, Articulated Models, Particle Swarm Optimization, Differential Evolution, Global Continuous Optimization

1. Introduction

Object detection is the task of finding a given object in an image or video sequence. In computer vision, this is a challenging task because of artefacts (due to the acquisition system, to the environment or to other optical effects like perspective, illumination changes, etc.) which may affect the features even of well-defined objects. These problems may cause the same objects to have different appearance depending on the specific conditions under which images are acquired. The task may become even harder in the presence of "interferences" such as partial occlusions and cluttered backgrounds. Finally, and possibly most importantly, the task is further complicated by the intrinsic variability exhibited by different instances of objects belonging

*Corresponding Author. Tel. +39 0521 905731
Email addresses: rob_ugo@ce.unipr.it (Roberto Ugolotti), nashed@ce.unipr.it (Youssef S. G. Nashed), pmesejo@ce.unipr.it (Pablo Mesejo), spela.ivekovic@strath.ac.uk (Špela Ivekovič), mussi@ce.unipr.it (Luca Mussi), cagnoni@ce.unipr.it (Stefano Cagnoni)

to large general categories or, even more relevantly, by living beings, or parts of their bodies.

Image retrieval, surveillance, medical image analysis and advanced driving assistance are only some of the most typical applications which rely on object detection and impose some strict requirements:

1. detection accuracy;
2. robustness with respect to noise (generically comprising all possible alterations described in the previous paragraph);
3. execution speed, up to real-time performances.

Inevitably, all object detection tasks, implicitly or explicitly, must rely on a model of the object that is to be detected. This is reflected in the main computer vision approaches, which can be divided into bottom-up and top-down approaches.

On the one hand, the more classical bottom-up (feature-based) approaches are based on processes where all significant generic low-level features (edges, textures, regions, etc.) are extracted from the image, and then "recomposed" into sets which may represent the target. These processes follow a "natural" strategy which mimics biological perception, in the absence of any expectations about what the object of perception may be. In this case, knowledge about the object (its model) is only applied after an exhaustive extraction of features from the image. In practice, this leads to a lack of specificity, as all image regions are usually explored independently of any expected content. This may further turn into a lack of accuracy or an increase in computation load, due to the huge number of possible outcomes which are taken into consideration.

On the other hand, top-down (model-based) approaches introduce knowledge about the object to be detected, as well as about the physical laws which both the object itself and the image acquisition process obey. Regardless of the nature of the object under consideration, a prototype model is created which describes its most important invariant features (like color, shape, relative position with respect to other structures, etc.). At the same time, knowledge about the physics of the object and the imaging process makes it possible to define a transformation, from the object space to the image space, which can represent all possible poses or deformations of the object as they may appear in the corresponding image. In this case, a hypothesis is made about object location, deformation and orientation and represents a point in the transformation domain. The hypothesis is then checked by evaluating the similarity between the expected appearance of the modeled object in the image and the actual content of the corresponding image region. This approach is generally preferable when two criteria are satisfied: reliable a priori knowledge must be available, which can limit the size of the transformation domain, and the implementation of the transformation must be fast enough to allow a search algorithm to efficiently explore such a domain. The problem is then turned into a global optimization problem.

In the last two decades, research on global optimization has been very active [1, 2], and many different deterministic and stochastic algorithms for continuous optimization have been developed. Among the stochastic approaches, population-based evolutionary algorithms (EAs) and swarm intelligence (SI) methods [3, 4] offer a number of advantages that make them attractive: easy implementation, implicit parallelism, robust and reliable performance, global search capability, no need of specific information about the problem to solve, good insensitivity to noise, and no requirement for a differentiable or continuous objective function.

In this paper, we focus onto two paradigmatic applications in which the requirements listed above are satisfied, which makes it possible to solve hard, real-world object detection problems by the joint use of a model-based approach and of swarm-intelligence/evolutionary methods: hippocampus localization in histological images of sections of mouse brains [5] and 3D body pose estimation from multiple views [6]. The first requirement is satisfied by two models, which are based on anatomical knowledge about the shape and appearance of the hippocampus and about the mechanical constraints which regulate the degrees of freedom of limb articulations in the human body, respectively. The second requirement is satisfied by

two metaheuristics, Particle Swarm Optimization (PSO) and Differential Evolution (DE), which are easily parallelizable and can therefore be used to implement a fast search within the transformation domain. To meet the third requirement, we have parallelized these optimization techniques, as well as the localization functions, for execution by Graphics Processing Units (GPU) within the nVIDIA™ CUDA environment [7]; we compare their results in terms of both quality and computational efficiency.

The paper is organized as follows: in section 2 we provide the theoretical foundations of our work, as well as an overview of previous related work. In section 3 we discuss model-based object detection. Section 4 describes the two problems taken into consideration. Section 5 describes how the two metaheuristics used to solve them were parallelized within CUDA. Section 6 compares the experimental results by appropriate statistical tests. Finally, section 7 includes some final remarks and a discussion about possible future developments.

2. Theoretical background and related work

This section shortly introduces the building blocks of the framework within which we are evaluating our approach. We assume readers of this journal to be mainly interested and knowledgeable about metaheuristics, while possibly not being aware of computer vision and pattern recognition techniques. Therefore, while the introduction to deformable models and templates aims at making our approach easier to understand by readers who are not familiar with these methods, the short introduction to PSO and DE aims at defining the terms and parameters related with the two algorithms, to which we will refer later on in the paper, and clarify which variants, among the myriad available for both algorithms, we have actually implemented.

The section is closed by a few words on the GPU programming environment CUDA by nVIDIA™, within which we have developed and implemented our methods, to help the readers' understanding of the implementation choices which will be described further on.

2.1. Deformable Models and Templates

The term "deformable models" generally refers to curves or surfaces, defined within the image domain, that are deformed under the influence of "internal" forces, related with the curve features, and "external" forces, related with the image regions surrounding the curve. Internal forces enforce regularity constraints and keep the model smooth during deformation, while external forces are defined such that the model is attracted toward an object or other features of interest within the image. The term "deformable models" was first used in the late eighties [8, 9], and one of the first examples, called "snakes" or Active Contour Models, was presented shortly after in [10].

Active Shape Models (ASMs) [11] add more prior knowledge to deformable models. These shape models derive a "point distribution model" from sets of labeled points (landmarks) selected by an expert in a training set of images; in each image, a point, or set of points, is placed on the part of the object corresponding to its label. The model considers the points' average positions and the main modes of variation found in the training set. While this kind of model has problems with unexpected shapes, since an instance of the model can only take into account deformations which appear in the training set, it is robust with respect to noise and image artefacts, like missing or damaged parts. Active Appearance Models [12] extend ASMs by considering not only the shape of the model, but also other image properties, like intensity or color.

Finally, "deformable templates" represent shapes as deformations of a given prototype or template. They have been successfully applied to object tracking [13] and object matching [14]. A deformable template is defined by a prototype which mathematically describes the prior knowledge about the object shape as the most likely appearance of the object being sought. Additionally, one needs to provide a mathematical description of the possible relationships between the template and all admissible object shapes, to represent the possible transformations which can deform the basic template and turn it into the target object as appears in the image. A template is specified by a set of parameters which control its deformation until it matches

the target.

Deformable templates can be considered a sub-class of properly-defined deformable models. In fact, the latter are active models, since they rely on a rigorous physical paradigm, according to which they finally relax into an equilibrium status, i.e., a configuration which minimizes the global energy related with the opposing action of external and internal forces. The transformations which are applied to template models, instead, can be assimilated with the action of external forces only, as the influence of image features is not directly represented in the transformations but only implicitly "externally" described by an error function which measures the difference between the deformed template and the object to be matched/detected.

Articulated models are a kind of deformable templates of particular interest for modeling the human body or modeling robotic arms in industrial robotics. In these models, the object is represented by a kinematic chain of segments and joints, or by a set of such chains connected to a common reference structure/segment. In three dimensions, each joint has up to three degrees of freedom, which can be represented, for example, by the relative rotation angles, around the reference axes, between the two segments connected by the joint under consideration. An example of this model is the skeleton of the body shown in the lower-left part of Figure 1.

Within this framework, many object detection tasks can be reformulated as global optimization problems in which a metaheuristic is used to optimize either the parameters of some active contour model, maximizing the accuracy of the relaxation process, or the parameters of a deformable template, which directly encode both its position and appearance within an image. In this paper we use the latter approach, within which, by exploring the parameter domain, we find an optimal configuration for the template by maximizing a function representative of the similarity between the model and a region of the image under investigation.

As mentioned before, we compare the results obtained by doing so using two different metaheuristics (PSO and DE) which are briefly introduced in the following paragraphs.

2.2. Particle Swarm Optimization

Particle Swarm Optimization is a bio-inspired optimization algorithm introduced by Kennedy and Eberhart [15]. It is based on the simulation of the social behavior of bird flocks. In the last fifteen years PSO has been applied to a very large variety of problems [16] and many variants of the original algorithm have been proposed [17].

During the execution of PSO, a set of particles moves within a function domain searching for the optimum of the function (best fitness value). The motion of the i^{th} ($i = 1, N_p$) particle can be described by the following two simple equations which regulate the particle's position and velocity:

$$\begin{aligned} v_i(t) &= w \cdot v_i(t-1) \\ &+ c_1 \cdot rand() \cdot (BP_i - P_i(t-1)) \\ &+ c_2 \cdot rand() \cdot (BGP - P_i(t-1)) \\ P_i(t) &= P_i(t-1) + v_i(t) \end{aligned}$$

where $v_i(t)$ and $P_i(t)$ are respectively the velocity and position of the particle in the present iteration, c_1, c_2 and w (inertia factor) are positive constants, $rand()$ returns random values uniformly distributed in $[0, 1]$, BP_i is the best-fitness position visited so far by the particle. In the basic algorithm "global-best PSO", BGP is the best-fitness position visited so far by any particle of the swarm. In several variants, termed "local-best PSO", the swarm is subdivided into particle neighborhoods which can assume different topologies. In that case, BGP becomes BGP_i and represents the best-fitness position visited so far by any particle in the i^{th} particle's neighborhood. Among the possible neighborhoods, the ring-shaped one is particularly interesting for its simple implementation, as well as for the role it may play in optimizing the efficiency of PSO parallelization [18] and even, sometimes, for the improvement of convergence speed [19] it may bring.

2.3. Differential Evolution

Differential Evolution, first introduced by Storn and Price [20], has recently been one of the most successful Evolutionary Algorithms for global continuous optimization, especially when the function to be optimized is multimodal and non-separable [21]. Unlike traditional EAs, DE perturbs the individuals of the current generation by the scaled differences of other randomly-selected and distinct individuals. Therefore, no separate probability distribution has to be used for generating the offspring [22]. This way, in the first iterations the population members are widely scattered in the search space and possess great exploration ability. During optimization, the individuals tend to concentrate in the regions of the search space with better values, so the search automatically focuses onto the most promising areas [23].

In DE, new individuals that will be part of the next generation are created by combining individuals that are already members of the current population. Every individual acts as a parent vector and, for each of them, a donor vector is created. In the basic version of DE, the donor vector for the i^{th} parent (X_i) is generated by combining three random and distinct individuals X_{r1}, X_{r2} and X_{r3}. The donor vector V_i is calculated by what is called mutation of difference vectors as follows:

$$V_i = X_{r1} + F \cdot (X_{r2} - X_{r3})$$

where F (scale factor) is a parameter that strongly influences DE's performances and typically lies in the interval [0.4, 1]. Recently, several mutation strategies have been applied to DE, experimenting with different base vectors and different numbers of vectors for perturbations. For example, the original method explained above is called DE/rand/1, which means that the first element of the donor vector equation X_{r1} is randomly chosen and only one difference vector (in our case $X_{r2} - X_{r3}$) is added. After mutation, every parent-donor pair generates a child (U_i), called trial vector, by means of a crossover operation.

$$U_{i,j} = \begin{cases} V_{i,j} & if(rand_{i,j} \leq C_r \text{ or } j = j_{rand}) \\ X_{i,j} & otherwise \end{cases}$$

As described in the above equation, the j^{th} component/dimension of the i^{th} donor vector is obtained by means of uniform (binomial) crossover, where $rand_{i,j}$ is a uniformly distributed random number in the range $[0, 1]$, C_r is the crossover rate, and j_{rand} is a randomly selected dimension. The newly-generated individual U_i is evaluated by comparing its fitness to its parent's fitness. The best survives and will be part of the next generation.

DE shares some features with swarm intelligence techniques, mainly related to the interaction among particles and the selection scheme. In particular, both DE and PSO are stochastic, population based, real-valued algorithms, and designed for challenging continuous optimization problems (non-differentiable, nonlinear and/or multimodal functions) using few control parameters. DE can also be considered as an Evolutionary Algorithm (EA), but differs from traditional EA algorithms in the aspect of generating new vectors by adding the weighted difference vector between two population members to a third member.

Despite several limitations which have been highlighted and the availability of other algorithms which perform better on global optimization benchmarks [24, 25], PSO and DE have recently become very popular. The main reason for their success is related to their associating good average performances with easy implementation. However, the feature which is most relevant to our work, and shared with other evolutionary and swarm intelligence algorithms, is the fact that, being population-based and featuring a limited dependency between each individual's operations, PSO and DE can be parallelized very easily. All the work described in this paper relies on GPU-based parallel implementations of PSO and DE, developed within the nVIDIA^{TM} CUDA environment, whose main features are briefly introduced below.

2.4. General-purpose GPU computing

Modern graphics hardware has gained an important role in the area of parallel computing. Graphic cards have been used in 3D graphics applications and gaming but, recently, they have also been more and more frequently used to accelerate numeric computation, in what is usually called general-purpose GPU (GPGPU) programming [26]. The main advantage of using GPUs lies in their structure: while standard CPUs usually contain a handful of complex computational cores, memory registers and large cache memory, GPUs contain up to several hundreds of cores grouped into so-called multiprocessors, organized such that each ALU of a multiprocessor executes the same operations on different data, stored in registers or device memory. In contrast with standard CPUs, which can reschedule operations (out-of-order execution), current GPUs are an example of an in-order architecture, but this drawback can be overcome by their massive parallelism, as described by Hager et al. [27], when the problem to be solved fits their features well.

CUDA (Compute Unified Distributed Architecture) is a GPGPU environment by nVIDIA^{TM} available for its most recent GPUs. The language used within CUDA, CUDA-C, is an extension of the C programming language which allows one to implement GPU-based parallel functions, called kernels. These are executed in parallel as a number of different CUDA threads. nVIDIA^{TM} terms this computation model Single Instruction Multiple Thread (SIMT) model. SIMT is very much like the Single Instruction Multiple Data (SIMD) computational model, with some differences in the management of conditionals. In this model, each core can execute a sequential thread. Cores are logically organized in groups; all cores in the same group execute the same instruction at the same time on different data. In doing so, they can benefit from GPU features such as fast shared memory, atomic data manipulation, and synchronization. The set of thread groups which execute a kernel is called, in CUDA terminology, a grid. A kernel can use several types of memory: fast local and shared memory, large but slow global memory, and fast read-only constant memory and texture memory.

2.5. Related work

The literature about object detection is very wide; in a huge number of computer vision applications a solution to this problem is required, and several different techniques have been developed to tackle it. We complete this introductory section with a brief review of some literature related with model-based object detection, considering only work in which evolutionary computation and swarm intelligence techniques have been used in applications similar to the ones described in this paper.

Most literature about the combination of EAs with deformable models deals with applications of Genetic Algorithms (GAs). In [28], a medical application is described where a GA evolves a population of shapes, using prior shape knowledge to produce feasible deformations while also controlling the scale and localization of these deformations. Following with medical imaging applications, in [29, 30], GAs are used to optimize the model presented by Kass [10]; in [31], Ghosh and Mitchell have used GAs to evolve a segmenting contour by incorporating both texture and shape information to extract objects without prominent edges, such as the prostate in pelvic computed tomography (CT) images. A two-stage GA-based active-model method has been proposed in [32], and applied to the segmentation of the lateral ventricles in magnetic resonance images of the brain. Finally, in [33], the authors have described a method for interactive segmentation of 3D medical images that relies on an edge detector and an elastic contour model for both of which the parameters are optimized by a GA.

More recently, Particle Swarm Optimization has been successfully used in conjunction with various types of deformable models. Asl and Seyedin [34] apply the technique proposed in [29] using PSO instead of a GA, obtaining similar results in terms of precision but in shorter time. Finally, in [35], a multi-population PSO is used to drive an active contour model, emphasizing the capability of the model to adapt to shapes with strong concavity.

Much less work is related with combining Differential Evolution and deformable models. An example, again related with medical imaging problems, is presented in [36]. In this work, DE, in combination with a greedy search algorithm, is used to evolve a discrete implementation of an elastic mesh, called Topological Active Nets, for CT image segmentation.

Finally, model-based object detection is tackled by GPU implementations of soft computing techniques in papers like [37], where CUDA is used for accelerating a tracking algorithm based on adaptive appearance

models and PSO. In [38], the GPU is used to implement a real-time full-body tracking algorithm, based on an articulated 3D body model, similar to or directly based on the method described in [39, 40], which was implemented only sequentially. Tracking is obtained by searching for a model configuration that best matches the observed human silhouette in the input images, and the parameter space is searched by a GPU-based implementation of Particle Swarm Optimization, where each particle represents a configuration of the model, and therefore a pose.

3. Object Detection using Deformable Models

As mentioned in the introduction, object detection algorithms can be divided into two large classes: more classical, low-level feature-based bottom-up approaches, and model-based top-down approaches. The object detection applications described in this paper belong to the latter. This is a very general class of methods which can be adapted to possibly any kind of objects, subject to the availability of invariant visual features (like color, shape, feasible and infeasible deformations ...), typical of the class of objects to detect, that can be represented by a set of parameters, as well as of knowledge about the physical laws which the object obeys.

The definition of the model can be achieved in two steps:

- selection of the most relevant intrinsic features of the object, based on the observation of a significant subset of object instances;
- definition of the ranges within which the features may vary, based on the knowledge of the physical laws which regulate the object and the image acquisition process.

In fact, the model must be able to account for all possible transformations (perspective effects, other geometric transformations, deformations due to noise, etc.) by which the object is affected during image acquisition. Finally, the choice of the features and parameters to be included in the model must also take into consideration environmental constraints, such as background complexity, presence of other objects which may generate occlusions, real-time constraints, etc.

Once the model is defined, a similarity measure must also be defined that drives the model search towards the actual configuration of the object, i.e., that reaches its maximum when the image representation of the model is perfectly superimposed to the object as it appears in the image.

The problem then becomes a global optimization problem, that is the search of the model parameters which maximize the similarity measure. The landscape searched by the optimization algorithm is usually strongly multimodal. Therefore, the next crucial step to be taken is the selection of a good heuristic, which can deal with such a rough landscape both efficiently and effectively.

The generic detection process, taking the body pose estimation problem as an example, is summarized in Figure 1. Regarding a given measure of quality (the overlap of the image projection of the model with the original image), this process iteratively tries to improve a population of candidate solutions, which represent deformations of the basic model. So, the search for the best model transformation is driven by the similarity function and by the metaheuristic used to optimize it. In each iteration, models are deformed and their image projections are superimposed onto the original image(s), in order to compute the degree of overlap with the target image(s), which determines the fitness value. The search is over when a stopping criterion is met.

4. Applications

This section describes the two real-world problems to which the general method presented above has been applied, to evaluate its performance and to compare the effectiveness of DE and PSO as search algorithms within this framework.

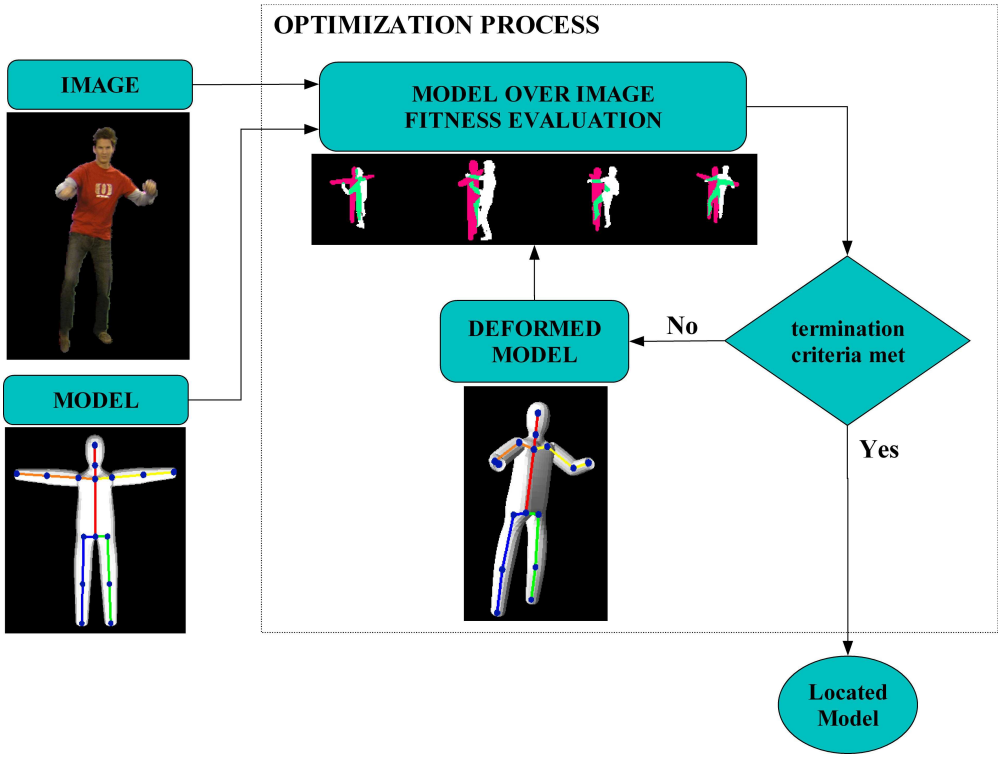


Figure 1: General optimization schema.

4.1. Human Body Pose Estimation

Three-dimensional human body pose estimation from video is the problem of extracting an accurate estimate of the posture of a human body, along with its location in space, from an image or a frame within a video stream. This is a complex problem that has been invariably formulated as a high-dimensional space search problem, due to the complexity of the human body pose parameterization. The problem has been tackled by trying to reduce the complexity of the search while also relying on effective search schemes.

The search complexity can be reduced based on local predictions, e.g., using particle filters [41, 42], or by partitioning the search space into smaller, more manageable subspaces [41, 43]. The use of machine learning techniques to define specific motion models for particular actions from training data collected in advance has also been considered [44, 45]. These approaches suffer from various setbacks. The particle-filtering solutions critically rely on a high number of particles to adequately represent the posterior distribution, which increases their computational complexity beyond practical use when considering a wide variety of motion. As well, relying on pre-trained motion models causes the human body tracking approaches to lose their generalization abilities, which points to methods that can reliably provide motion estimates without depending on much prior knowledge [46].

In [39, 40], an effective search algorithm was proposed, which is capable of recovering the pose without any prior knowledge of the nature of motion. The main drawback of the method is its huge computational complexity, which makes the time required for execution of a standard sequential implementation hardly acceptable. However, relying on the parallel nature of both the search algorithm and the multi-view pose estimation problem by implementing the approach on a graphical processing unit (GPU), the authors showed that they could reach execution times acceptable for practical purposes [6].

The input consists of N views of the body, taken from different angles. From each image, we extract the silhouette of the body, i.e., a binary image in which all pixels belonging to the body are set to 1. The set of silhouettes represents the target to be matched to the silhouettes generated by a transformation of the

model, according to the following steps:

- a pose estimation is generated by the search algorithm;
- a 3D rendering of the body, in such a pose, is made;
- a set of N images, corresponding to the projections of the rendered body (silhouettes) on the image planes of the input cameras, is computed.

The body model consists of two layers, the skeleton and the skin. The skeleton layer is defined as a set of homogeneous 4×4 transformation matrices which encode the information about the position and orientation of every joint with respect to its parent joint in the kinematic tree hierarchy. The skin layer, which represents the second layer in the model, is connected to the skeleton through the joints' local coordinate systems. Each joint controls a certain area of the skin. Whenever a joint or limb moves, the corresponding part of the skin moves and deforms with it. As the skin is a subdivision surface, only the base mesh has to be specified in the corresponding joint coordinate system. After the joint configuration has been specified, the base mesh is subdivided by repeatedly applying the Catmull-Clark subdivision operator [47] until the desired smooth shape of the body is obtained.

Considering the body composed of head, torso, and a three-joint kinematic chain for each limb, the model has 32 degrees of freedom, represented by real-valued parameters: three of them represent the global body position in space, while the other 29 represent relative angles, in space, between consecutive segments of the kinematic chains, i.e., joint orientations. These are subject to anatomical constraints which limit both their number and possible value range. A more detailed description of the model can be found in [6].

The matching function compares the silhouettes extracted from the original images to the silhouettes generated by the model in its candidate pose. Let the images containing the *original* silhouettes be denoted as I_i^o, $i = 1...N$. Similarly, let I_i^m, $i = 1...N$ denote the images of the *model* silhouettes. The cost function can then be written as follows:

$$E = \sum_{i=1}^{N} \frac{1}{Z_i} \sum_{1}^{row} \sum_{1}^{col} (I_i^o \ \& \ I_i^m), \tag{1}$$

where *row* and *col* denote the number of image rows and columns, respectively, and & denotes the bitwise *AND* operation. Coefficients Z_i are the normalization constants obtained by counting the number of silhouette pixels in every original image. Therefore, the fitness value that can be obtained for each view ranges from 0 to 1, with 0 corresponding to the absence of overlapping between the two silhouettes, and 1 to a perfect overlap. Thus, the overall fitness value (E) ranges from 0 to N.

4.2. Hippocampus Localization in Histological Images

Among the large number of applications of automatic localization and segmentation methods in clinical and experimental medicine, there is great interest in automated methods to accurately, robustly, and reproducibly localize the hippocampus in brain images after discoveries which established its role as an early biomarker for Alzheimer's disease and epilepsy [48, 49].

We consider the problem of extracting the region where the hippocampus is located from the mouse brain images in the Allen Brain Atlas (ABA) [50], a huge, publicly available image database, which has recently provided scientists with a gene-expression map for future study and investigation. The ABA contains a genome-scale collection of histological images (cellular resolution gene-expression profiles) obtained by in situ hibridization (ISH) of serial sections of mouse brains.

Figure 2 shows some examples of images of the hippocampus extracted from the ABA. As can be seen, the common feature shared by all hippocampi are the two main parts (called sg and sp), whose shape is not too variable, which are usually characterized by lower intensity values and higher color saturation with respect to the surrounding structures.

The template model used for detection has been built based on these considerations. The model comprises two elements which can be considered template models themselves, one for sg and one for sp (see Figure 4).

Figure 2: Examples of hippocampus images.

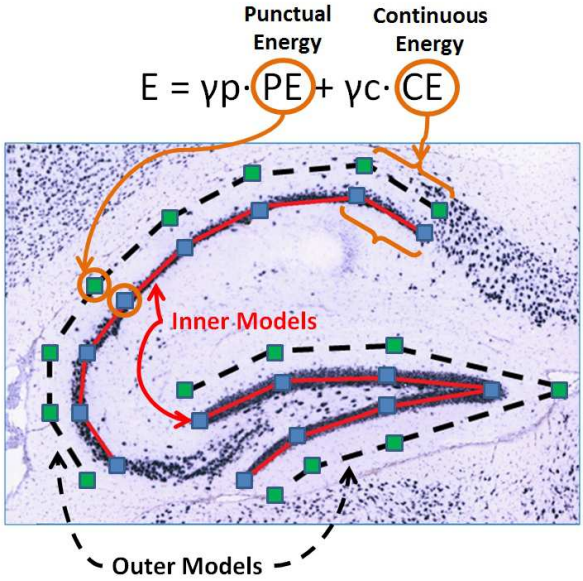


Figure 3: Inner and outer models superimposed to the hippocampus image, to clarify the definition of the terms PE and CE.

Similarly to body pose estimation, the parameters of the models do not refer to the absolute positions of the points, but describe the relative positions (or shifts) between consecutive points in polar coordinates, even if, in this case, in two dimensions. For every segment that composes the model, a prototype and a range of acceptable values of lengths and angles has been determined based on the empirical analysis of a set of training images.

Every template model is, in turn, composed of two parts: an "inner set" of points that lies precisely on the anatomical part we want to locate, and an "outer set" of points that lies just outside it, obtained by shifting the points of the inner set. Accordingly, optimizing the matching function must attain the following goals [5]:

- minimize the intensities of the pixels that belong to the inner set;
- maximize the intensities of the pixels that belong to the outer set;
- keep the model shape as similar as possible to the prototype (see Figure 3).

Here, as in classical deformable models, we consider the model to be subjected to external forces (corresponding to the constraints on image intensity) and internal forces (corresponding to the constraints on the model shape).

The matching function M to be maximized is composed by three terms:

$$M = E - (I + C)$$

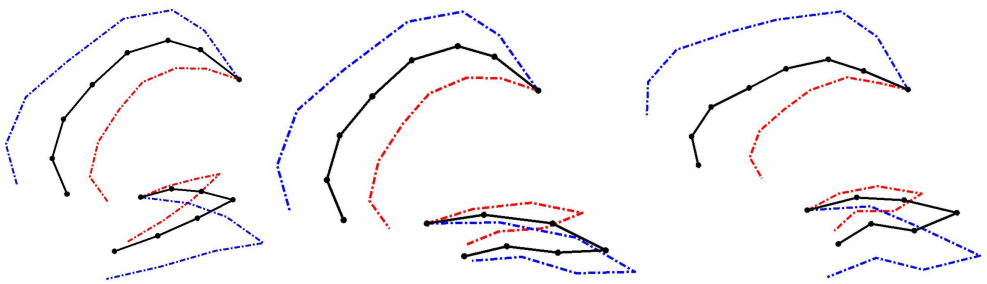


Figure 4: Examples of hippocampus models. The black solid line represents the prototype of the model, the blue and red dotted lines represent the lower and upper limits for the possible deformations of the model.

Searching the best match for the first term, E, moves and deforms the model in order to minimize the intensity of the pixels corresponding to the inner set, and maximize the intensity of the pixels corresponding to the outer set. For both sets, we evaluate the intensity of the image within 3×3 neighborhoods, $N_3(P_i)$, of all points P_i in the model (PE) and in p intermediate points that lie on the line segment between two consecutive ones (CE). The intensities of the points in the inner and outer sets are subtracted, both in PE and CE, originating the term E according to the following equations:

$$\begin{aligned} PE &= \sum_{i=1}^{n} [T(N_3(I_i)) - T(N_3(O_i))] \\ CE &= \sum_{i=2}^{n} \sum_{j=1}^{p} T(I_{i-1} + \frac{j}{p+1}(I_i - I_{i-1})) \\ &- \sum_{i=2}^{n} \sum_{j=1}^{p} T(O_{i-1} + \frac{j}{p+1}(O_i - O_{i-1})) \\ E &= \gamma_P \cdot PE + \gamma_C \cdot CE \end{aligned}$$

where n is the number of points in the model, I_i and O_i represent the i^{th} points of the inner and of the outer model, respectively, while $T(P)$ is the intensity of the image in P if P is a point, or the average intensity if P is a neighborhood. The value p has been set to 20 and the weights γ_P and γ_C have been empirically set to 5 and 1, respectively.

The term I tends to limit the deformation of the model and is computed as:

$$I = \xi_\rho \cdot \sqrt{\sum_{i=2}^{n} (\rho_i - \rho_{mi})^2} + \xi_\vartheta \cdot \sqrt{\sum_{i=2}^{n} (\vartheta_i - \vartheta_{mi})^2}$$

where ξ_ρ and ξ_ϑ are two positive parameters that weight the deformation ability of the model, n is the number of points of the model, ρ_{mi} and ϑ_{mi} the angle and length of the i^{th} element of the prototype model, respectively, and ρ_i and ϑ_i are the corresponding terms for the model under evaluation.

The last term, C, regulates the deformability of the model using the distance between the first and the last point of the inner model:

$$C = \xi_c \cdot \|I_n - I_1\|$$

If $\xi_c < 0$ the two extremes of the model repel each other, if $\xi_c > 0$ they attract each other.

For a more detailed description of this matching function, please refer to [5].

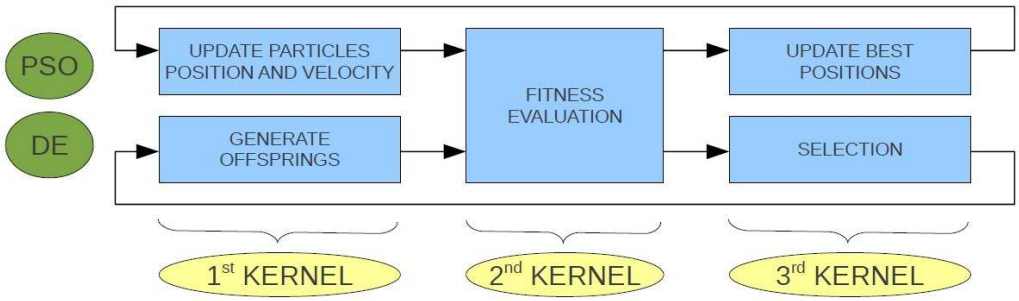


Figure 5: CUDA kernels making up our PSO and DE implementations.

5. CUDA Implementation

Swarm intelligence techniques are intrinsically parallel, because every swarm member has few dependencies on the others, so all operations aimed at adapting an individual's values, like position update or fitness evaluation, can be executed with few (or none at all) interactions with the other elements of the swarm.

From this point of view, in PSO, the only data to be shared among particles is the best position visited so far by any member of the swarm or of a particle's neighborhood. We implemented a standard PSO with particles organized in a ring topology of radius equal to one, as described in [18]. Therefore, each particle needs only to be aware of the fitness values of its two neighbors. This choice has been made because this topology is known to work well with multimodal problems, besides further reducing the dependencies between particles.

Our CUDA-based implementations of PSO and DE share the same structure, based on three CUDA kernels, whose operations are described in Figure 5. The kernels that execute the operations on the particles of PSO and DE have a similar structure, in which a thread block performs one particle's updates, while each single thread manages the update along one problem dimension. The structure of the kernels responsible for fitness evaluation depends, obviously, on the function itself: the main advantage of the parallel version of the hippocampus localization function is that all accesses to the image under consideration are made in parallel for all points of all particles of the swarm and the values thus obtained are added up using parallel reductions.

In the body pose estimation, the fitness function is defined as the number of overlapping pixel pairs in the silhouettes derived from the input frames and in the corresponding projections of the deformed model. Since we use a large number of views from as many cameras (ten in the experiments reported), each of which has a different projection/calibration matrix, a sequential pixel-by-pixel comparison would be excessively time consuming. Therefore, we take advantage of the GPU computing power to compare the pixels from each input image to the corresponding pixels from the image encoded by each swarm member. The image representing the hypothesized pose according to the model is divided into sections that are handled by CUDA thread blocks, then further sub-divided into pixel rows, processed by one thread, to count the overlapping pixels in parallel. Finally, the final fitness of a candidate solution is obtained by adding up the accumulated count vectors using parallel reductions.

6. Experimental Results

In this section we show some results of the tests we performed on the two problems we took into consideration. For both applications, we first show the results obtained on a simplified version of the problem, for which a ground truth is available, and then the results of the tests on actual images.

Tests were run on a computer equipped with a 64-bit Intel® Core i7 CPU running at 2.80 GHz with 6 Gb of RAM using CUDA v. 4.1 on a nVIDIA™ GeForce GTS450 graphics card with 1GB of DDR memory and compute capability 2.1.

We compared the results obtained by our CUDA implementations of DE and PSO. The parameter values used in the tests were set starting from the most commonly used values reported in literature, and refined during the development of the system. The values we set for the most relevant parameters are shown in Tables 1 and 5.

6.1. Human Body Pose Estimation

Our algorithms were tested on a set of 4 test sequences, kindly made available by the CVSSP, University of Surrey. They were acquired in a dedicated multi-camera acquisition studio and consist of 10 synchronized videoclips with resolution 720×576, and a frame rate of 25 fps.

DE	PSO
$Cr = 0.9$	$c_1 = 2.0$
$F = 0.5$	$c_2 = 2.0$
Uniform Crossover	$w = 2.0/e^x$
Mutation: DE/rand/1	
Population Size = 10	Population Size = 10

Table 1: Parameters used for human body pose estimation. Regarding the inertia factor w, we set $x_{initial} = 2.0$; $x = x + 0.05$ if, at the end of a generation, the global best has not improved. For the very first frame $x_{initial}$ was set to 1.0 to increase the algorithm's exploration ability, when it is required to recover the initial pose from scratch.

Since these sequences come with no ground truth, we decided to create a "synthetic" sequence to statistically estimate the error made by our system in recovering the pose of the body. To do so, we took the sequence containing the most complex (and fastest) movements, which represents a man performing a karate kick, and let our system optimize it multiple times for a very high number of generations. After collecting the best results (highest fitness values) for each frame, we rendered the silhouette images of our model in those very same positions. This way we obtained an artificially created sequence of which the articulated model we employ exactly matches all the silhouettes available and for which we know, frame by frame, the actual pose of each joint of the model. In other words, we created a synthetic sequence which comes with "ground truth" values for all the parameters we need to optimize. After this, we compared the three-dimensional position of every joint of the model in the reference sequence and the values obtained as output by the test runs of our method.

It is important to remark that, in the final tests, instead of setting a fixed number of iterations/generations as in most iterative algorithms, we used the value of the decreasing inertia parameter defined in Table 1 as a stopping criterion for both DE and PSO, ending our process when w fell below 0.1.

The first two tables refer to the results obtained processing the reference sequence. In particular, in Table 2 we show the results obtained by PSO and DE, expressed as distances, and in Table 3 as fitness values (higher fitness values are associated with better solutions). In Table 4 we show the global results as fitness values computed on the other four sequences.

The first column in all tables is the mean value of the measure under consideration over all runs and frames. For example, in Table 3, the value in the first row and first column is the fitness obtained by DE averaged over the 500 executions of the algorithm (50 frames and 10 runs). The second column reports the mean of the average standard deviations obtained for every joint in the model. The third and fourth columns report the mean of the worst and best values, respectively, averaged over all frames in each run. The fifth column is the mean of the median values for each run. Finally, the last column in all tables reports the p-value obtained with the Wilcoxon Signed-Rank test [51] with a significance level of 0.001.

The null hypothesis used in Table 2 was that the median of distances obtained by PSO is greater or equal than the median of the distances obtained by DE. In Tables 3 and 4 the p-value refers to the following

null hypothesis: the median fitness obtained by PSO is less or equal than the median fitness obtained by DE.

In Figures 8 and 9, all the results obtained, per joint and per frame respectively, are plotted.

	Average	StdDev	Worst	Best	Median	Wilcoxon
DE	6.41	6.60	41.42	0.36	3.76	$< 1.0E-10$
PSO	4.32	4.47	32.71	0.22	2.40	-

Table 2: Results of human body pose estimation: average distance values (in cm) to the joints obtained processing the reference sequence in ten independent runs.

	Average	StdDev	Worst	Best	Median	Wilcoxon
DE	8.21	0.11	8.06	8.42	8.91	$< 1.0E-10$
PSO	8.24	0.13	8.05	8.45	8.98	-

Table 3: Results of human body pose estimation: average fitness values obtained processing the reference sequence in ten independent runs.

	Average	StdDev	Worst	Best	Median	Wilcoxon
DE	8.30	0.11	8.13	8.49	8.89	$< 1.0E-10$
PSO	8.34	0.12	8.15	8.52	8.94	-

Table 4: Results of human body pose estimation: average fitness values obtained processing all the "real" video sequences in ten independent runs.

In the first frame, the initialization of the swarm is completely random, while in the subsequent ones the swarm is initialized in a vicinity of the best pose found in the previous frame, thereby implementing some sort of tracking. As the pose changes only slightly between two consecutive frames, performing the optimization over the whole search space is both unnecessary and time consuming. However, to investigate the general localization ability of DE and PSO, the optimization was allowed to run for more iterations, 500 in this case, and the hierarchical optimization steps, described in [6], were also removed to increase the complexity of the problem. Results are reported in Figure 7, showing how fitness values improve during the optimization process. As explained before, only the results obtained processing the first frame are actually representative of the global search ability of the method used.

As the tables show, the results obtained by PSO in this problem using the hierarchical optimization are better than the ones obtained with DE, in terms of fitnesses and distances. However, if we increase the complexity of the problem by removing the hierarchical strategy and the time constraints, optimizing all the parameters at the same time, DE obtains better results than PSO (see Figure 7). This behavior can be explained, in first place, taking into account that DE is one of the best-performing methods for large-scale continuous optimization problems [22]; therefore, the more complex the environment the better the expected performance with respect to other methods. A second explanation can be the evolutionary nature of DE; since the scaled differences of randomly selected and distinct population members are combined to create new solutions, the weighted combination of good partial solutions can produce very good global results.

It is also important to notice that the best results are obtained using the hierarchical approach, in which, whenever a good position for one part of the body is found, all other joints are constrained to this newly found best position (i.e. they cannot explore other orientations and positions that are inconsistent with it). With respect to execution time, the hierarchical version of the human body pose estimation using DE takes on average 4677.60 ms per frame, while the corresponding version based on PSO takes 4810.33 ms.

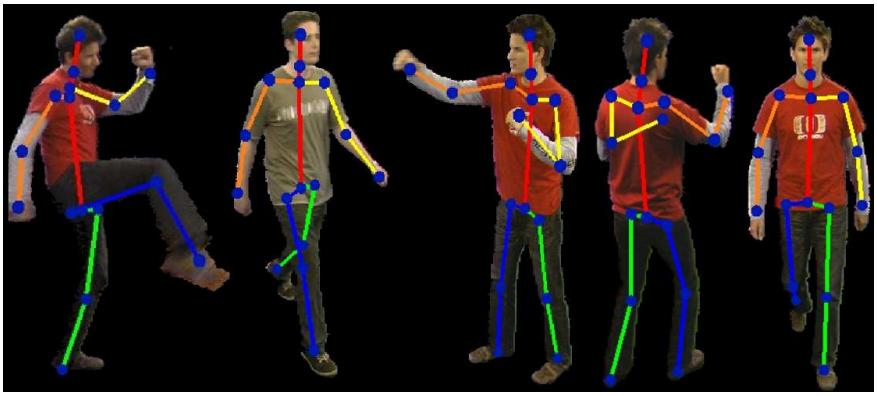


Figure 6: Examples of pose estimation results. Each of the five human figures corresponds to one of the five real video sequences analyzed.

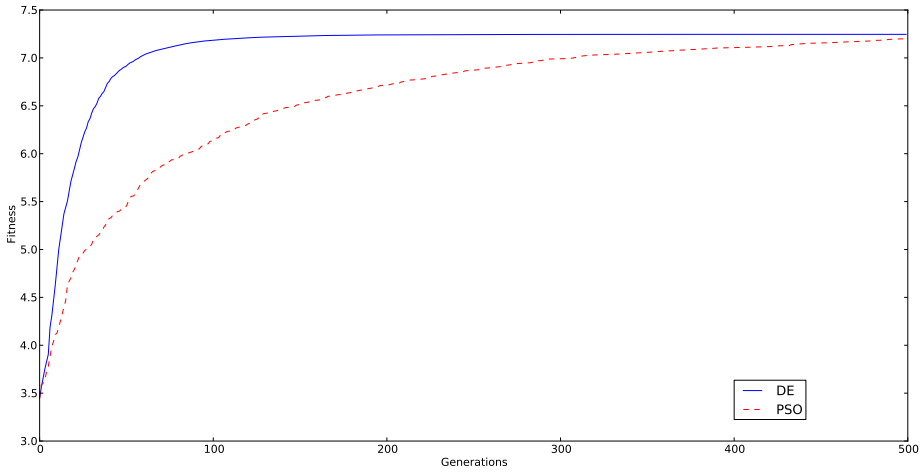
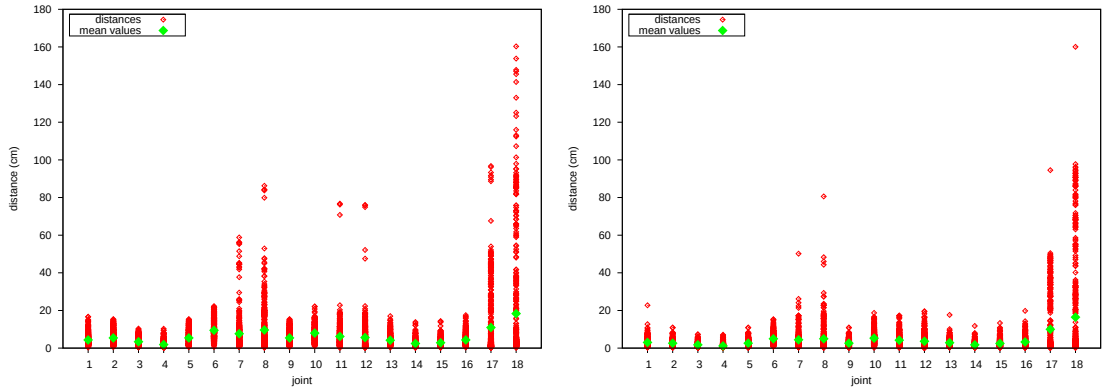
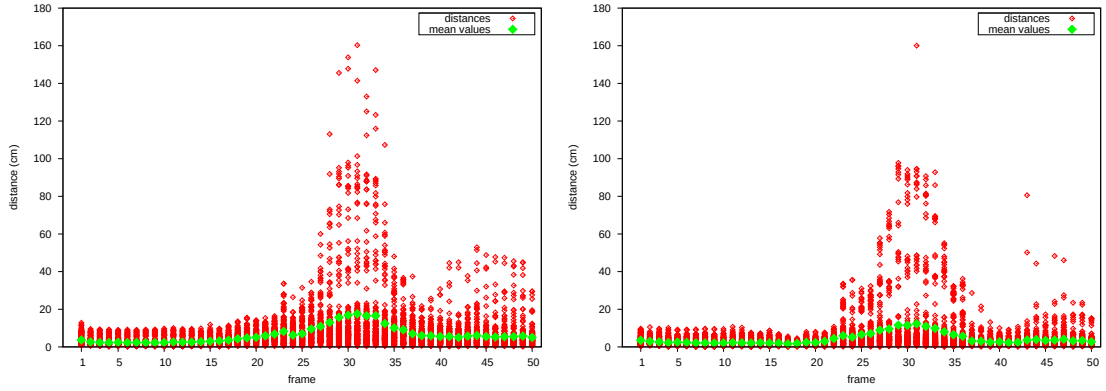


Figure 7: Average fitness values vs number of generations for PSO and DE in the body pose estimation problem for the first frame of the 5 sequences. Full optimization of the whole body.



(a) DE (b) PSO

Figure 8: Body pose estimation: per-joint performance on the reference video sequence. Scatter plot of the distances (in cm) of each joint from the ground truth estimated over all frames over 10 runs. Means are represented by lighter bullets.



(a) DE (b) PSO

Figure 9: Body pose estimation: per-joint performance on the reference video sequence. Scatter plot of the distances from the ground truth of all joint estimates over 10 runs for each of the 50 frames. Means are represented by lighter bullets.

(a) Original Image (b) Segmented Image (c) Model over Segmented Image

Figure 10: Simplified version of the hippocampus: (a) the original image of an hippocampus, (b) the simplified version obtained by manual segmentation by an expert, (c) the computed model superimposed to the previous image.

6.2. Hippocampus Localization in Histological Images

As for body pose estimation, we first tested our approach on a simplified version of the problem. We created 15 "clean" images of hippocampi by manually segmenting the anatomical structures we wanted to match and removing everything else in the image (see Figure 10); a ground truth was computed based on the skeleton of these segmentations. In order to avoid erroneous or incomplete manual segmentations, this step was supervised by an expert in molecular biology.

DE	PSO
$Cr = 0.9$	$c_1 = 1.19$
$F = 0.7$	$c_2 = 1.19$
Uniform Crossover	$w = 0.72$
Mutation: DE/rand/1	
Population Size = 32	Population Size = 32
Iterations = 500	Iterations = 500

Table 5: Parameters used in hippocampus localization.

We evaluated the results in terms of average distance, in pixels, between the ground truth and the final position of all points (7 for sg, 8 for sp) in the model, and in terms of fitness values. Table 6 shows the comparison between PSO and DE on the 15 test images (500 independent runs for each image) for both parts of the model (sp and sg). This table has the same format and reports the same data as Table 3. Table 7 shows the same results in terms of fitness values (as Table 4).

Figures 11 and 12 show the scattered plot of the results obtained by DE and PSO on sp and sg, respectively. Each figure shows the results obtained by one of the two techniques on each test image as distances between the ground truth points and the ones obtained by the localization algorithm. Figure 13 shows the results of hippocampus localization on simulated histological images, as well as real ones taken from the ABA.

According to Tables 6 and 7, DE performs better in the sp region, as the statistical tests confirm. Regarding sg, DE provides better average fitness values but, when distance is considered, there is no statistical evidence of a difference between PSO and DE. The standard deviation of the results obtained on sp by DE is greater than the one obtained by PSO, while the opposite happens with sg. So, each algorithm shows a more stable behavior in different parts of the hippocampus. DE always achieves better average results in sp but with higher standard deviation. In sg the results obtained by DE and PSO are very similar, but the standard deviation of DE is smaller.

In Table 6, some data, like the average worst obtained by DE in sp, are probably affected by some outliers. It is important to remind that we are reporting extreme values; in fact, the average median distance for each image (over 500 runs), is 6.47 pixels from the ground truth. In fact, only in 7 runs, over the 7500 total, the distance exceeded 50 pixels from the ground truth. For an absolute evaluation of these results, it is also to be noticed that we are considering very high resolution images (about $1\mu m$ per pixel).

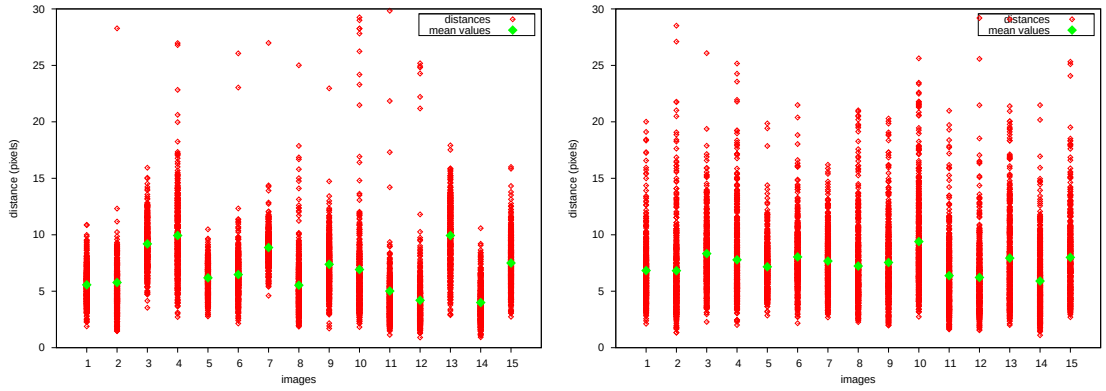
		Average	StdDev	Worst	Best	Median	Wilcoxon
sp	DE	6.83	3.80	62.57	2.21	6.47	-
	PSO	7.42	3.28	28.24	2.09	6.83	$< 1.0E-10$
sg	DE	7.42	1.74	12.73	3.95	7.23	0.03
	PSO	7.39	2.15	14.2	3.37	7.18	-

Table 6: Results of hippocampus localization over manually segmented images: distances.

In the subsequent test phase, we randomly selected from the Allen Brain Atlas 320 images, mapping the expression of 320 different genes. This selected subset is representative of the different hippocampus shapes and contrasts which characterize images of different genes. For each optimization technique, we ran 100 tests for every image, for a total of 32000 experiments. Results are summarized in Table 8 using the

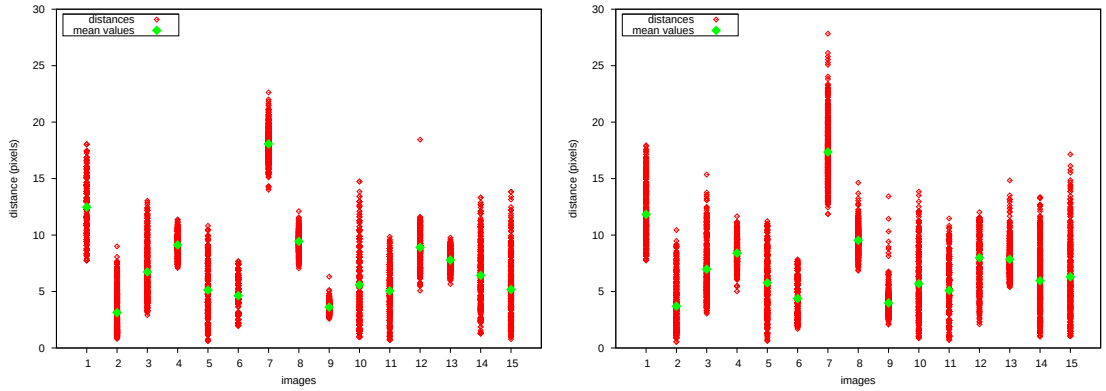
		Average	StdDev	Worst	Best	Median	Wilcoxon
sp	DE	149.59	2.37	114.36	150.54	149.91	-
	PSO	148.84	2.02	127.88	150.69	149.37	$< 1.0E-10$
sg	DE	158.95	0.04	158.61	159.03	158.96	-
	PSO	158.93	0.10	158.04	159.07	158.94	$< 1.0E-10$

Table 7: Results of hippocampus localization over manually segmented images: fitness values.



(a) DE (b) PSO

Figure 11: Hippocampus localization: performance on the images segmented manually. Scatter plot of the distances (in pixels) between each point of the ground truth and the sp model over the 15 images, performing 500 runs for every image. Some of the outliers are not represented in the plots for a better visualization. Means are represented by lighter bullets.



(a) DE (b) PSO

Figure 12: Hippocampus localization: performance on the images segmented manually. Scatter plot of the distances (in pixels) between each point of the ground truth and the sg model over the 15 images, performing 500 runs for every image. Means are represented by lighter bullets.

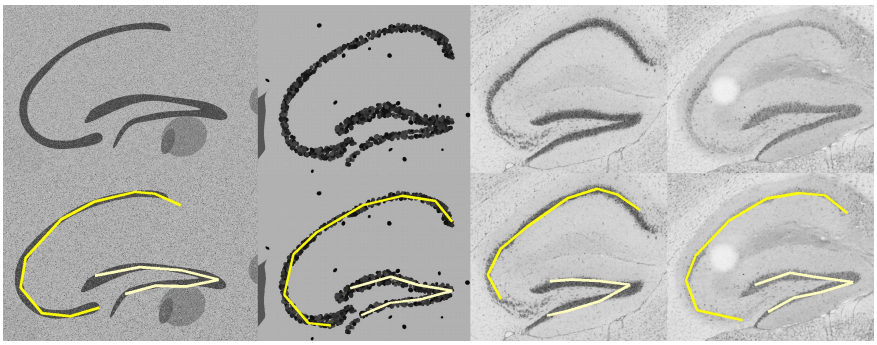


Figure 13: Examples of hippocampus localization on synthetic (left) and real images (right).

same format as for the other tables. Since there is no ground truth for these images, a comparison can only be made between the fitness values. It can be observed how, in this more difficult task (real images with cluttered background), DE significantly overcomes PSO, showing better robustness with respect to noise. A possible reason for this is that, while every coordinate of PSO is updated independently of the others, DE with a high crossover value (Cr in Table 5), is able to evolve its elements with better "coordination" between the different dimensions. Figure 14 shows another interesting aspect: PSO reaches very good values in few iterations, but DE overcomes it when the number of iterations becomes higher.

		Average	StdDev	Worst	Best	Median	Wilcoxon
sp	DE	127.02	15.42	81.10	139.37	132.13	-
	PSO	119.09	13.30	72.92	136.83	121.93	$< 1.0E-10$
sg	DE	148.88	1.83	138.75	150.81	149.21	-
	PSO	145.04	3.44	132.06	150.27	145.56	$< 1.0E-10$

Table 8: Results of hippocampus localization in real images: fitness values.

As shown in Table 8, DE achieved the best average fitness, the best average worst results, the best average best results, in localizing both sp and sg.

To assess the significance of this result, we performed a statistical test with a confidence level of 0.001. The paired Wilcoxon signed-rank test was performed, assuming as null hypothesis that the median of PSO fitness is greater than or equal to the median of DE. The corresponding p-values (shown in the last column of Table 8) reject, in all cases, the null hypothesis, showing that significant differences exist between the performance of Differential Evolution and PSO. Moreover, the average worst values show that DE avoids local minima and is able to obtain good transformation parameters more often than PSO.

Regarding execution time, the hippocampus localization using DE takes on average 28.59 μs, while PSO takes 26.07 μs.

7. Discussion and Conclusions

We presented a general method for object detection in images based on deformable models and swarm intelligence/evolutionary optimization algorithms. This approach can be theoretically applied to any kind of objects, as long as some invariant features exist that can be described by a parametric model which embeds suitable constraints, in order to create feasible transformations. To search the optimal transformation, DE and PSO were used and implemented as parallel algorithms within the nVIDIA^{TM}CUDA environment for faster execution.

We compared the results of the two metaheuristics over two different object detection problems: human body pose estimation in video sequences and hippocampus localization in histological images. In one of

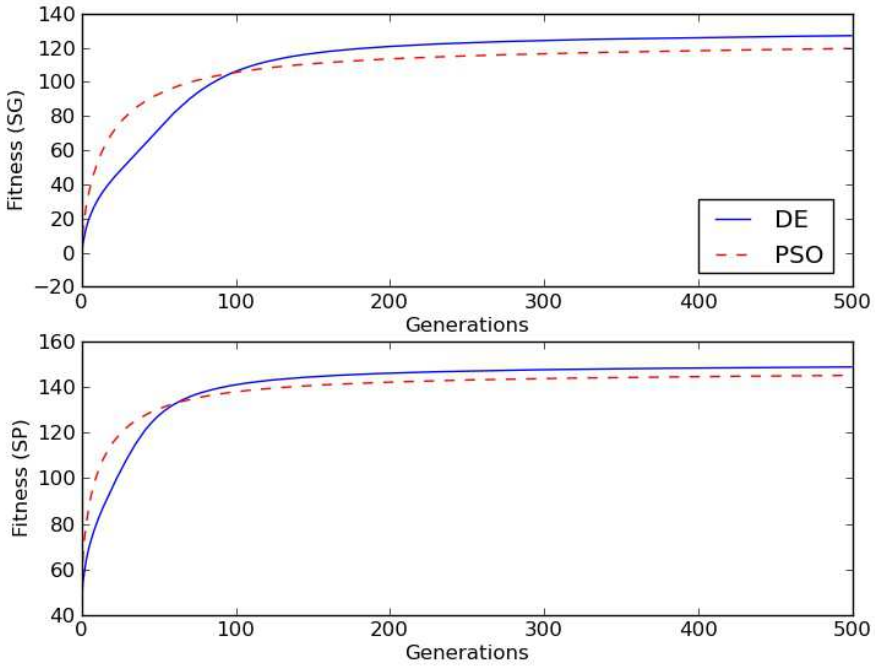


Figure 14: Fitness values vs number of generations for PSO and DE in the hippocampus localization problem for the 320 real images.

these two problems, DE obtained better results than PSO, in terms of accuracy, while, in the other one, PSO was better according to the same criteria. As the No Free Lunch theorem [52] demonstrates at the most general scale, we could not prove the general superiority of one method over the other even on a limited set of two rather similar applications.

Despite this, some global conclusions can be drawn regarding the different behaviors of the two meta-heuristics. PSO appears to suffer from stagnation, obtaining good results quickly but not reaching as good results as DE in the long term. In fact, in more complex problems with more variables to optimize, and allowing a greater number of iterations, DE has shown greater effectiveness in finding optimal solutions than PSO, as happens in the body pose estimation without restrictions, where all parameters are optimized at the same time. However, if good results are needed in few generations, PSO seems more appropriate given its ability to quickly find good solutions. In order to improve quality and efficiency of DE, a plausible alternative could be the inclusion of the concept of super-fit control adaptation in DE frameworks [53], where DE is hybridized with three meta-heuristics, each having different roles and features (PSO, that supports DE in the beginning of the optimization process by helping to generate a super-fit individual, and two local search methods to refine the results).

Furthermore, as a possible justification of the results, it is important to notice that PSO performs a directional search, i.e. it allows a high performance for unimodal or non-strongly multimodal landscapes, as the one that is, supposedly, generated by the body pose estimation problem using the hierarchical optimization. On the contrary, DE has more exploratory properties and is more efficient when multiple basins of attraction with similar strength are present in the landscape, like in the hippocampus localization, that is a highly multimodal problem, as shown in [54].

Regarding future developments related to hippocampus localization, we expect to develop and apply proper segmentation techniques for this anatomical structure within the image regions located as described in this paper. Eventually, this segmentation will allow the accurate detection of specific hippocampus regions where gene expression can be analyzed and characterized based on visual features. Other approaches will also be tested, such as multi-population approaches, to locate the two structures within the hippocampus at the same time and not independently of each other as happens now.

In any case, the main aim of this work was to present a general object detection framework where deformable models are optimized by means of two well-known metaheuristics, and not to find the best configuration of parameters for these population-based optimization techniques. For this reason, more DE mutation/crossover operators or PSO topologies, as well as more sophisticated DE/PSO implementations, like the improved jDE [55] or the scale factor inheritance by Weber et al. [56], were not employed and will be the subject of some of our future work.

Thanks to the GPU-based parallelization, we will be able to make massive tests on thousands of genes in reasonable time. This will allow us to analyze quantitative information about the general levels and the spatial distribution in the brain of all the mammalian genes, which is the final goal of the project within which our application was developed.

Regarding body pose estimation, we are currently extending its implementation to make it possible to run larger swarms and use fewer camera views, as well as testing more sophisticated fitness functions. Another possible interesting improvement could be the use of increasingly popular 3D sensors, such as cheap depth cameras, instead of the specialized multi-camera setup, for easier acquisition of image which can dramatically extend the usability of our approach.

Acknowledgments

Roberto Ugolotti is funded by Compagnia di S.Paolo, Torino, Italy, within the "Neuroscience" programme.

Youssef S. G. Nashed and Pablo Mesejo are funded by the European Commission (MIBISOC Marie Curie Initial Training Network, FP7 PEOPLE-ITN-2008, GA n. 238819).

The authors would like to thank Prof. A. Hilton from the CVSSP, University of Surrey, for providing the test sequences, Mr A. Patney from University of California, Davis, for sharing his CUDA implementation of

the Catmull-Clark subdivision, and the anonymous referees for their valuable comments that allowed them to highly improve the paper quality.

All the mouse brain images were downloaded from the website: Allen Mouse Brain Atlas [http://mouse.brain-map.org]. Seattle (WA): Allen Institute for Brain Science. ©2011.

References

[1] C. Floudas, C. Gounaris, A review of recent advances in global optimization, Journal of Global Optimization 45 (2009) 3–38.
[2] A. Zhigljavsky, A. Zilinskas, Stochastic global optimization, Springer, 2007.
[3] A. E. Eiben, J. E. Smith, Introduction to Evolutionary Computing, Springer, 2003.
[4] A. P. Engelbrecht, Computational Intelligence: An Introduction, Wiley Publishing, 2nd edn., 2007.
[5] R. Ugolotti, P. Mesejo, S. Cagnoni, M. Giacobini, F. Di Cunto, Automatic Hippocampus Localization in Histological Images using PSO-Based Deformable Models, in: Proc. Genetic and Evolutionary Computation Conference, GECCO '11 (Companion), 2011.
[6] L. Mussi, Š. Ivekovič, S. Cagnoni, Markerless Articulated Human Body Tracking from Multi-View Video with GPU-PSO, in: G. Tempesti, A. M. Tyrrell, J. F. Miller (Eds.), Evolvable Systems: From Biology to Hardware - 9th International Conference, ICES 2010 York, UK, September 2010 Proceedings, 195–207, 2010.
[7] nVIDIA, nVIDIA CUDA programming guide v. 4.0, nVIDIA Corporation, 2011.
[8] D. Terzopoulos, K. Fleischer, Deformable Models, The Visual Computer 4 (1988) 306–331.
[9] D. Terzopoulos, A. Witkin, M. Kass, Constraints on Deformable Models: Recovering 3D Shape and Nonrigid Motion, Artificial Intelligence 36 (1988) 91–123.
[10] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active contour models, Int J of Computer Vision 1 (1988) 321–331.
[11] T. F. Cootes, C. J. Taylor, D. H. Cooper, J. Graham, Active shape models-their training and application, Comput. Vis. Image Underst. 61 (1995) 38–59.
[12] T. F. Cootes, G. J. Edwards, C. J. Taylor, Active Appearance Models, in: Proceedings of the European Conference on Computer Vision, vol. 2, 484–498, 1998.
[13] Y. Zhong, A. K. Jain, Object tracking using deformable templates, IEEE Trans. on Pattern Analysis and Machine Intelligence 22 (2000) 544–549.
[14] A. K. Jain, Y. Zhong, S. Lakshmanan, Object Matching Using Deformable Templates, IEEE Trans. on Pattern Analysis and Machine Intelligence 18 (1996) 267–278.
[15] J. Kennedy, R. Eberhart, Particle Swarm Optimization, in: Proc. of IEEE International Conference on Neural Networks, ICNN'95, vol. 4, 1942–1948, 1995.
[16] R. Poli, Analysis of the publications on the applications of Particle Swarm Optimisation, J. Artificial Evolution and Applications (2008) 1–10.
[17] A. Banks, J. Vincent, C. Anyakoha, A review of Particle Swarm Optimization. Part I: background and development, Natural Computing 6 (2007) 467–484.
[18] L. Mussi, F. Daolio, S. Cagnoni, Evaluation of parallel Particle Swarm Optimization algorithms within the CUDA architecture, Information Sciences 181 (2011) 4642–4657.
[19] L. Mussi, Y. S. Nashed, S. Cagnoni, GPU-based asynchronous Particle Swarm Optimization, in: Proceedings of the 13th annual conference on Genetic and evolutionary computation, GECCO '11, 1555–1562, 2011.
[20] R. Storn, K. Price, Differential Evolution- A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, Tech. Rep., International Computer Science Institute, 1995.
[21] J. Vesterstrom, R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, in: IEEE Congress on Evolutionary Computation, CEC2004, 1980–1987, 2004.
[22] S. Das, P. Suganthan, Differential Evolution: A Survey of the State-of-the-Art, IEEE Transactions on Evolutionary Computation 15 (2011) 4–31.
[23] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artif. Intell. Rev. 33 (2010) 61–106.
[24] N. Hansen, S. Finck, R. Ros, A. Auger, Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions, Research Report, INRIA, 2009.
[25] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, A. Auger, PSO Facing Non-Separable and Ill-Conditioned Problems, Research Report RR-6447, INRIA, 2008.
[26] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krger, A. E. Lefohn, T. J. Purcell, A Survey of General-Purpose Computation on Graphics Hardware, Computer Graphics Forum 26 (2007) 80–113.
[27] G. Hager, T. Zeiser, G. Wellein, Data access optimizations for highly threaded multi-core CPUs with multiple memory controllers, in: IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008., 1–7, 2008.
[28] C. McIntosh, G. Hamarneh, Evolutionary Deformable Models for Medical Image Segmentation: A Genetic Algorithm Approach to Optimizing Learned, Intuitive, and Localized Medial-Based Shape Deformation, in: S. Smith, S. Cagnoni (Eds.), Genetic and Evolutionary Computation: Medical Applications, John Wiley & Sons Ltd, 46–67, 2010.
[29] L. MacEachern, T. Manku, Genetic algorithms for active contour optimization, in: Proc. IEEE International Symposium on Circuits and Systems, ISCAS '98, vol. 4, 229–232, 1998.

[30] L. Ballerini, Genetic snakes for color images segmentation, in: Applications of Evolutionary Computing, vol. 2037 of *Lecture Notes in Computer Science*, Springer, 268–277, 2001.
[31] P. Ghosh, M. Mitchell, Segmentation of medical images using a genetic algorithm, in: Proc. 8th annual conference on Genetic and evolutionary computation, GECCO '06, 1171–1178, 2006.
[32] Y. Fan, T. Jiang, D. Evans, Volumetric segmentation of brain images using parallel genetic algorithms, IEEE Transactions on Medical Imaging 21 (2002) 904–909.
[33] S. Cagnoni, A. B. Dobrzeniecki, R. Poli, J. C. Yanch, Genetic algorithm-based interactive segmentation of 3D medical images, Image and Vision Computing 17 (1999) 881–895.
[34] M. Asl, S. Seyedin, Active Contour Optimization using Particle Swarm Optimizer, in: Information and Communication Technologies, ICTTA '06, vol. 1, 1522–1523, 2006.
[35] C.-C. Tseng, J.-G. Hsieh, J.-H. Jeng, Active contour model via multi-population particle swarm optimization, Expert Systems with Applications 36 (2009) 5348–5352.
[36] J. Novo, J. Santos, M. G. Penedo, Optimization of topological active nets with differential evolution, in: Proc. of the 10th International Conference on Adaptive and Natural Computing Algorithms, ICANNGA'11, 350–360, 2011.
[37] B. Rymut, B. Kwolek, GPU-supported object tracking using adaptive appearance models and Particle Swarm Optimization, in: Proceedings of the 2010 international conference on Computer vision and graphics: Part II, ICCVG'10, Springer-Verlag, Berlin, Heidelberg, 227–234, 2010.
[38] T. Krzeszowski, B. Kwolek, K. W. Wojciechowski, GPU-Accelerated Tracking of the Motion of 3D Articulated Figure, in: L. Bolc, R. Tadeusiewicz, L. J. Chmielewski, K. W. Wojciechowski (Eds.), ICCVG (1), vol. 6374, 155–162, 2010.
[39] Š. Ivekovič, E. Trucco, Y. R. Petillot, Human body pose estimation with particle swarm optimisation, Evol. Comput. 16 (2008) 509–528.
[40] V. John, E. Trucco, Š. Ivekovič, Markerless human articulated tracking using hierarchical particle swarm optimisation, Image Vision Comput. 28 (2010) 1530–1547.
[41] J. Bandouch, F. Engstler, M. Beetz, Evaluation of hierarchical sampling strategies in 3D human pose estimation, in: Proceedings of the 19th British Machine Vision Conference (BMVC), 2008.
[42] J. Deutscher, I. Reid, Articulated Body Motion Capture by Stochastic Search, Int. J. Comput. Vision 61 (2005) 185–205.
[43] J. Maccormick, M. Isard, Partitioned Sampling, Articulated Objects, and interface-quality hand tracking, 2000.
[44] F. Caillette, A. Galata, T. Howard, Real-time 3-D human body tracking using learnt models of behaviour, Comput. Vis. Image Underst. 109 (2008) 112–125.
[45] R. Urtasun, D. J. Fleet, A. Hertzmann, P. Fua, Priors for people tracking from small training sets, in: Proc. of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, 403–410, 2005.
[46] J. Gall, B. Rosenhahn, T. Brox, H.-P. Seidel, Optimization and Filtering for Human Motion Capture, Int. J. Comput. Vision 87 (2010) 75–92.
[47] J. Warren, S. Schaefer, A factored approach to subdivision surfaces, Computer Graphics and Applications 24(3) (2004) 74–81.
[48] J. Barnes, J. W. Bartlett, L. A. van de Pol, C. T. Loy, R. I. Scahill, C. Frost, P. Thompson, N. C. Fox, A meta-analysis of hippocampal atrophy rates in Alzheimer's disease, Neurobiology of Aging 30 (2009) 1711–1723.
[49] R. D. Terry, P. Davies, Dementia of the Alzheimer Type, Annual Review of Neuroscience 3 (1980) 77–95.
[50] Allen Institute for Brain Science, Allen Reference Atlases, `http://mouse.brain-map.org`, 2004-2006.
[51] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin (1945) 80–83.
[52] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Trans. on Evolutionary Computation 1 (1) (1997) 67 –82.
[53] A. Caponio, F. Neri, V. Tirronen, Super-fit control adaptation in memetic differential evolution frameworks, Soft Computing 13 (8-9) (2009) 811–831.
[54] S. Cagnoni, O. Cordón, P. Mesejo, Y. Nashed, R. Ugolotti, First Results and Future Developments of the MIBISOC Project in the IBISlab of the University of Parma, in: Proc. Genetic and Evolutionary Computation Conference, GECCO '12 (Companion), in press, 2012.
[55] J. Brest, B. Bošković, S. Greiner, V. Žumer, M. Maučec, Performance comparison of self-adaptive and adaptive differential evolution algorithms, Soft Computing-A Fusion of Foundations, Methodologies and Applications 11 (7) (2007) 617–629.
[56] M. Weber, V. Tirronen, F. Neri, Scale factor inheritance mechanism in distributed differential evolution, Soft Computing 14 (2010) 1187–1207.