Brain Cine MRI Segmentation Based on a Multiagent Algorithm for Dynamic Continuous Optimization

Julien Lepagnot, Amir Nakib, Hamouche Oulhadj, and Patrick Siarry
Laboratoire Images, Signaux et Systèmes Intelligents, LISSI, E.A. 3956
Université de Paris-Est Créteil
61 avenue du Général de Gaulle, 94010 Créteil, France
E-mail: {julien.lepagnot, nakib, oulhadj, siarry}@u-pec.fr

***Abstract*—In this paper, we propose a multiagent based evolution strategy algorithm, called CMADO, to evaluate the amplitudes of the deformations of the walls of the third cerebral ventricle on a brain cine-MR imaging. CMADO based segmentation technique is applied on a 2D+t dataset to detect the contours of the region of interest (i.e. lamina terminalis). Then, the successive segmented contours are matched using a procedure of global alignment. Finally, local measurements of deformations are derived from the previously determined matched contours. The validation step is realized by comparing our results to the measurements achieved on the same patients through a manual segmentation provided by an expert using Ethovision ®software.**

I. Introduction

Recently, optimization in dynamic environments has attracted a growing interest, due to its practical relevance. Almost all real-world problems are time dependent or dynamic, i.e. their cost function changes over time or the same operation is repeated periodically. For dynamic environments, the goal is not only to locate the optimum, but to follow it as closely as possible. Then, a dynamic optimization problem can be expressed by:

$$\begin{array}{ll} max & f(\vec{x},t) \\ s.t. & h_j(\vec{x},t) = 0 \; for \; j = 1,2,...,p \\ & g_k(\vec{x},t) \leqslant 0 \; for \; k = 1,2,...,l \\ & \vec{x} = [x_1, x_2, ..., x_n] \end{array} \tag{1}$$

where $f(\vec{x},t)$ is the objective function of the problem, $h_j(\vec{x},t)$ denotes the j^{th} equality constraint and $g_k(\vec{x},t)$ denotes the k^{th} inequality constraint. Both of them may change over time, denoted by t. In this paper, we focus on a dynamic optimization problem with time constant constraints. The main approaches to deal with DOPs can be classified into the following five groups [1]:

1) Generating diversity after a change: when a change in the environment is detected, explicit actions are taken to increase diversity and to facilitate the shift to the new optimum.
2) Maintaining diversity throughout the run: convergence is avoided all the time and it is hoped that a spread-out population can adapt to change more efficiently.
3) *Memory-based approaches*: these methods are supplied with a memory to be able to recall useful information from the past. In practice, they store good solutions in order to reuse them when a change is detected.
4) *Multipopulation approaches*: dividing up the population into several subpopulations, distributed on different optima, allows to track multiple optima simultaneously and increases the probability of finding new ones.
5) *Future prediction*: recently, another kind of methods, trying to predict future changes, has attracted much attention [2], [3], [4]. This approach is based on the fact that, in a real problem, changes can follow some pattern that could be learned.

In the literature, most proposed algorithms for dynamic optimization are population-based metaheuristics, which are global search, generally bio-inspired, algorithms. One can classify them in four categories: evolutionary algorithms (EAs), particule swarm optimization (PSO), ant colony optimization (ACO) and hybrid methods. For more details about the description of the dynamic metaheuristics that have been proposed in the literature in each of these categories, see [5]. In order to obtain a good performance on dynamic problems, some authors have tried hybrid methods, like in [6], [7], that propose hybrid PSO/EAs algorithms. However, the improvement of the performances is not significant.

Single solution based metaheuristics have also been applied to dynamic optimization, i.e. in [8] the authors proposed an algorithm that is based on extremal optimization (EO). EO mutates a single solution. This algorithm allows to obtain good results on the *Moving Peaks Benchmark* (MPB) [9]. However, this method is especially developed and fitted for this benchmark, and it seems not applicable to real world problems. In this paper, we propose a new version of the dynamic optimization algorithm proposed in [5]. This new version, called CMADO, enhances the performances of the original MADO (MultiAgent Dynamic Optimization) algorithm by using a covariance matrix. It is a multiagent method, that makes use of a population of agents to explore the search space. It is based on the following considerations: when optimizing in a multimodal environment, we need to keep track of each local optimum, to overcome the case where the global optimum *jumps* from one of them to another. The found optima are archived in a memory. Then, this memory can be used when a change is detected.

It is common that real world problems have a time costly evaluation of fitness functions. Hence, the computational cost of the proposed algorithm should be made as short as possible and expressed in terms of number of evaluations. The proposed algorithm makes use of the following inspirations:

- keeping information about the previous positions allows to prevent from wasting evaluations in unpromising zones of the search space;
- using a sampling of candidate solutions that optimally cover the local landscape may reduce the number of evaluations per trajectory step, without decreasing performance. For this goal, we maximize the distances between all the candidate solutions, whereas keeping them constricted into the current local landscape. More precisely, it is done by evaluating the candidate solutions on the surface of an hypersphere centered on the current position;
- the sampling of candidate solutions is adaptable to the local landscape. This is done by using an adaptable radius for the above-mentioned hypersphere;
- an exclusion radius for the current solution of each search trajectory is used to avoid the case where some search trajectories explore the same area of the search space.

In this paper, we propose to evaluate the pulsatile movement of the third cerebral ventricle using a process based on CMADO. The dataset is composed of cine-MRI sequences of 20 images that correspond to an R-R cardiac cycle [10]. An example of two images extracted from the sequence is presented on Fig. 1. The proposed method is based on two steps: the first step consists in a segmentation of the images using CMADO. In the second step, the successive couples of contours are geometrically matched by minimizing a quadratic distance between the points of the contours of two successive images. The rest of this paper is organized as follows. In section 2, the dynamic optimization problem at hand is presented. Section 3 describes the optimization algorithm. Experimental results are discussed in section 4. Conclusion and works in progress are in section 5.

II. The problem at hand

The segmentation problem has received a great deal of attention, thus any attempt to survey the literature would be too space-consuming. The most popular segmentation methods may be found in [11]. The common class of parametric methods used in brain MRI segmentation is based on an expectation-maximization framework. This class of methods is based on the assumption that a mixture Gaussian distribution is assumed as a model for the voxel intensity probability distribution. However, in most cases, the distribution is far from being Gaussian. Many authors tried to overcome this problem by regularizing the misclassification error through spatially constraining the segmentation process with prior information from a probabilistic atlas [12]. However, the method becomes very sensitive to the correct alignment of the atlas with the image and too time consuming. Actually, users (doctors) do not want to spend a lot of time waiting the result of segmentation, because of the large number of subjects. That induces the need for a fast segmentation algorithm. Many authors have applied to brain MRI classical segmentation methods, details are given in [13]. In order to overcome the problem of these methods, some post-classification methods were proposed [13], [14]. Actually, in order to segment a sequence of images, the authors restart the optimization process for each image, which is too time consuming. The main contribution of the work we present here is to show the importance of the use of dynamic optimization algorithms for brain cine MRI segmentation.

Dynamic image segmentation criterion

Before using this criterion we must fit the histogram of the image to be segmented to a sum of Gaussian probability density functions (pdf's). This procedure is named Gaussian curve fitting, more details about it are given below. The pdf model must be fitted to the image histogram, typically by using the maximum likelihood or mean-squared error approach, in order to find the optimal threshold(s). For the multimodal histogram h(i) of an image, where i is the gray level, we fit h(i) to a sum of d probability density functions [13]. The case where the Gaussian pdf's are used is defined by:

$$p\left(x\right)=\sum_{i=1}^{d}P_i\exp\left[-\frac{\left(x-\mu_i\right)^2}{\sigma_i^2}\right] \tag{2}$$

where P_i is the amplitude of Gaussian pdf on μ_i, μ_i is the mean and σ_i^2 is the variance of mode i, and d is the number of Gaussians used to approximate the original histogram and corresponds to the number of segmentation classes. Given an image histogram $h(j)$ (observed probability of gray level j), it can be defined as follows:

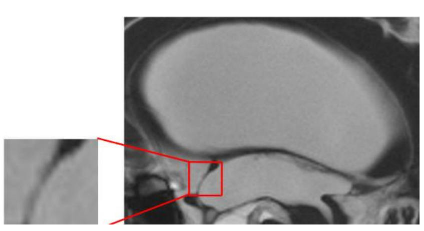
$$h(j)=\frac{g\left(j\right)}{\sum_{i=0}^{L-1}g\left(i\right)} \tag{3}$$

where $g(j)$ denotes the occurrence of gray-level j over a given image ranges $[0, L-1]$. Our goal is to find a vector of parameters, Θ, that minimizes the fitting error J, given by the following expression [6], [7]:

$$\text{Minimize}\, J=\sum_i\left|h(i)-p(\Theta,x_i)\right|^2 \tag{4}$$

where i ranges over the bins in the measured histogram. Here, J is the objective function to be minimized with respect to Θ, a set of parameters defining the Gaussian pdf's and the probabilities, given by $\Theta=\{P_i,\ \mu_i,\ \sigma_i;\ \ i=1,\,2,\cdots,\,d\}$. After fitting the multimodal histogram, the optimal threshold could be determined by minimizing the overall probability of error, for two adjacent Gaussian pdf's, given by :

$$e(T_i)=P_i\int_{-\infty}^{T_i}p_i\left(x\right)dx+P_{i+1}\int_{T_i+1}^{\infty}p_{i+1}\left(x\right)dx \tag{5}$$


(a)

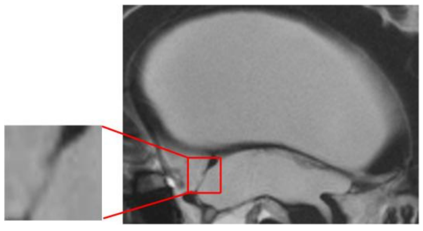

(b)

Fig. 1. Two images from a brain MRI sequence. (a) first image of the sequence, (b) 6th image of the sequence. Each sequence is composed of 20 images.

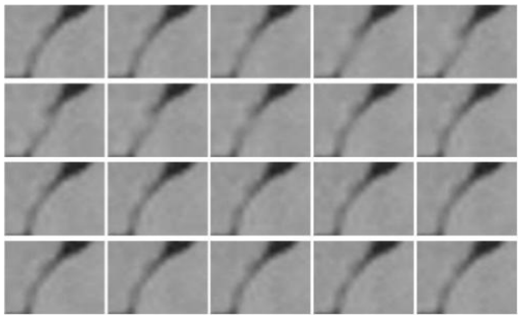


Fig. 2. An example of sequence of 20 cine-MR brain images about an entire cardiac cycle R-R.

with respect to the threshold T_i, where $p_i(x)$ is the i^{th} pdf and $i = 1, \ldots, d-1$. Then the overall probability to minimize is:

$$E(T) = \sum_{i=1}^{d-1} e(T_i) \tag{6}$$

where T is the vector of thresholds: $0 < T_1 < T_2 < \ldots < T_{(d-1)} < L - 1$. In our case L is equal to 256. The criterion in (4) has to be minimized for each image, as we are in the case of a sequence, then the fitting criterion becomes:

$$\text{Minimize } J(t) = \sum_i |h(i,t) - p(\Theta(t), x_i)|^2 \tag{7}$$

where t is the number of the current image in the cine MRI sequence, $\Theta(t)$ is same as Θ defined before but here is dependent on the image. $h(i,t)$ is the histogram of the image number t. In order to have an idea about the movement, we propose to state the segmentation problem in a 3D space. We build a 3D pseudo-volume by piling the images (Fig. 2) and we directly apply a dynamic optimization based segmentation procedure on the 2D+t cine-MRI dataset, in order to detect the contours of the selected zone. The procedure used to analyse the movements of the walls of the brain ventricles is given in Fig. 3.

III. The CMADO algorithm

A. Global description

MADO is a multiagent algorithm that consists in the following three modules:

1) Memory module: This module is used to archive the found optima. Indeed, a dynamic optimization method needs to keep track of each local optimum found, since one of them can become later the new global optimum.
2) Agent manager: This module manages the execution, creation, and deletion of agents. Agents perform local search, they fly from their current position to a better one, in their neighborhood, until they cannot improve their current solution, reaching thus a local optimum.
3) Coordinator: This module has a global vision of the strategy performed by every agent, and it guides them to search in promising zones of the search space. The coordinator supervises the whole search, and manages the interactions between memory and agent modules.

The overall scheme of the MADO and CMADO algorithms is illustrated in Fig. 4. As it is shown, all the interactions between the memory manager and the agent manager are through the coordinator. Moreover, all global decisions are taken by the coordinator. The memory manager maintains the archive of local optima, that are provided by the coordinator. The agent manager informs the coordinator about the found optima, and receives its instructions for creating, deleting, and repositioning agents.

B. Agent management module (the local search agents)

The local search agents explore the search space until they reach a local optimum. As agents are nearsighted, they can only test candidate solutions for their next move in a delimited zone of the search space, centered on their current position. This zone must be bounded. Without any information about the local landscape of an agent, to keep small the number of fitness function evaluations, we sample optimally the local landscape. The neighborhood of the agents is defined as a set of N candidate solutions placed on the hypersphere of

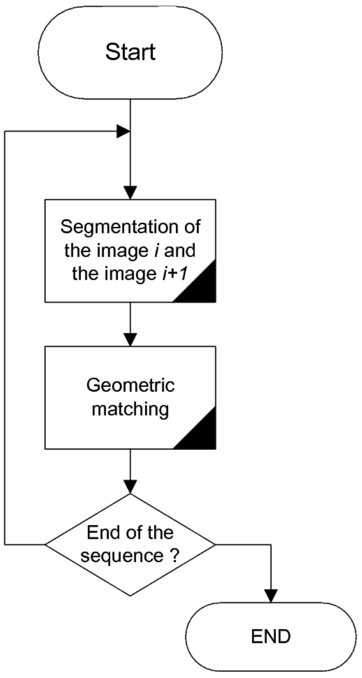


Fig. 3. Proposed procedure.

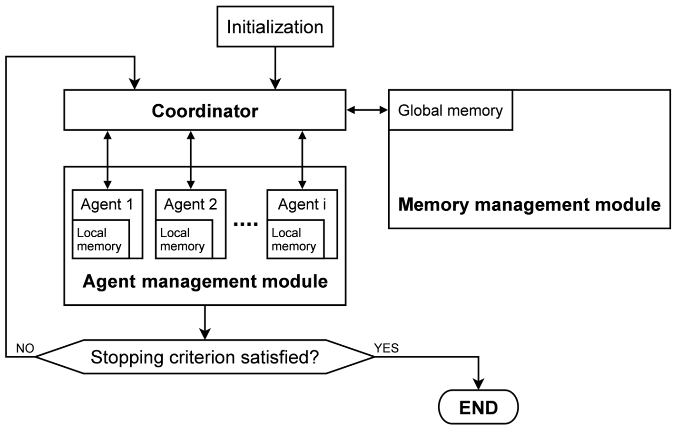


Fig. 4. Overall scheme of MADO algorithm.

radius R centered on the current position of an agent (N is a parameter of the algorithm). Fig 5 illustrates this neighborhood in dimension 2. Another set of N points, denoted by S, is generated and stored at the beginning of the MADO algorithm. These points are calculated in order to meet the following constraints:

- the smallest distance between them is maximized;
- the distances between them and the origin of the space (the point $(0, 0, \cdots, 0)$) are equal to 1;
- the number of dimensions of the space in which they are defined match the number of dimensions of the search space.

This computation is done using the well known electrostatic repulsion heuristic [15], that considers points as charged

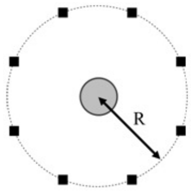


Fig. 5. The sampling of candidate solutions of an agent, when $d = 2$ and $N = 8$. The agent is depicted by a grey-filled circle. Candidate solutions are represented by black squares.

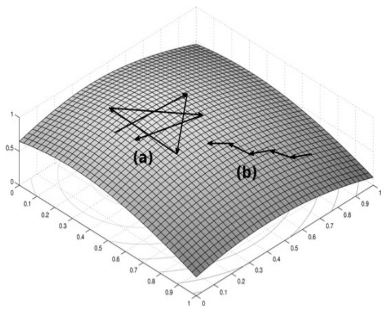


Fig. 6. The two kinds of trajectories that lead to a step size adaptation: (a) agent turning around an optimum, (b) agent hill climbing a large peak.

particles repelling each other. Starting from an arbitrary distribution, it uses point repulsion, where all points are considered to repel each other according to a $1/r^2$ force law, with r the distance between two points, and dynamics are simulated. The algorithm runs until the satisfaction of a stopping criterion, and the resulting set of points is returned. We use a stopping criterion that is satisfied when no point can be moved by a distance greater than ϵ. Then, this set of points is used by agents to get the positions of the candidate solutions of their current neighborhood.

The adaptation of the step size (the radius R) makes use of trajectory information gathered along the steps of an agent. The used *cumulative path length control based on variance-covariance matrix* was inspired from that described in [16], a low computation complexity version is used. The figure 6 illustrates the two possible kinds of *bad* trajectories that an agent may follow. If one of these cases is detected, then an increase or a decrease of the step size R is performed. The first case (Fig. 6(a)) may mean that the agent is turning around an optimum, without being able to reach it directly. This is due to a too large step size, leading to back and forth displacements, which cancel each other out. Thus, in this first case, a decrease in R is needed. On the contrary, the displacements on Fig. 6(b) are oriented in the same direction, which may mean that the agent is hill climbing a large peak. Then, to avoid too slow moves to the top of this peak, an increase in R is performed. When an agent has found a local optimum, the agent manager sends this optimum to the coordinator, that will transmit it to the memory manager. Then, the coordinator shows to the agent manager where this agent will be repositioned, in order to perform a new trajectory search. A new value is given to the step size (R) of this agent. One can remark that in a flat landscape, agents will not move from their initial position (their trajectory will consist in a succession of the same initial point), and stop their local search after δ_t steps, where δ_t is a parameter of MADO. Thus, in a flat landscape, the local optima transmitted to the memory manager are always the initial positions of the agents. In this paper, a new version of MADO, called CMADO, is proposed. Here, we propose to use a covariance matrix to adapt the neighborhood of the agents to the local landscape of the objective function. This strategy can improve the performance of MADO on ill-conditioned problems. This strategy is inspired from the CMA-ES (Covariance Matrix Adaptation Evolution Strategy) algorithm [16]. It consists in adapting a covariance matrix, denoted by C, at each step (displacement) of an agent, along with the adaptation of the cumulative dot product. Thus, each agent is equipped with its own covariance matrix, initialized to the identity matrix at the creation or at the relocation of the agent. It is adapted using the following equations (where A^T denotes the transpose of a matrix A) :

$$p = \frac{N}{2 \times d} \tag{8}$$

where N is the number of candidate solution that forms the neighborhood of an agent, and d is the dimension of the search space. Then, p enables the computation of the coefficient c_{cov} :

$$c_{cov} = \frac{p \ \times c_r + 1}{p + 1} \tag{9}$$

where c_r is the step size adaptation coefficient of an agent. Another coefficient, denoted by c_{id}, is calculated as follows :

$$c_{id} = (1 - c_r)\left(1 - \left|\vec{v}_{\max}^T \times \vec{D}\right|\right) \tag{10}$$

where $\vec{v}_{\max}$ is the normalized eigen vector associated to the highest eigen value, from the eigen decomposition of C calculated at the previous adaptation of C, and $\vec{D}$ is the current normalized displacement vector of the agent. Afterwards, the update of C is performed using the coefficient c_{cov} and c_{id}, as follows :

$$C' = I \times c_{id} + \left(C \times c_{\text{cov}} + D \times D^T(1 - c_{\text{cov}})\right)(1 - c_{id}) \tag{11}$$

where I is the identity matrix, and C' is used to update C. It is performed by replacing C by $C'/(eigen'_{max})$, where $eigen'_{max}$ is the highest eigen value from the eigen decomposition of C' (then, the highest eigen value from the eigen decomposition of C is equal to 1). Afterwards, the eigen values $eigen_0, eigen_1, ..., eigen_d$ and the normalized eigen vectors $\vec{v}_0, \vec{v}_1, ..., \vec{v}_d$ are updated from the eigen decomposition of C.

The eigen values and the normalized eigen vectors of the covariance matrix are then used to compute a set $S_a = \vec{s}'_1, \vec{s}'_2, ..., \vec{s}'_N$ of points from the precalculated set $S = \vec{s}_1, \vec{s}_2, ..., \vec{s}_N$ of uniformly spaced points. The set S_a is used to compute the candidate solutions of the agent at the next step of its local search (instead of using the set S, as it is done in the original version of MADO). It is performed according to the following equations :

$$B = \left[\ \sqrt{eigen_0} \times \vec{v}_0 \quad \cdots \quad \sqrt{eigen_d} \times \vec{v}_d\ \right] \tag{12}$$

Thus, each column of the matrix B is a vector in the basis used to compute the candidate solutions $\vec{s}'_i$:

$$\vec{s}'_i = B \times \vec{s}_i \tag{13}$$

C. Memory management module

The memory manager maintains the archive of local optima found by the agents. This archive must be bounded, its size is fixed by a predefined number n_m of entries. Thus, all the found optima cannot be included into this archive, only the best ones will be stored. When the archive is full, some decisions are taken looking to the situation, more details are given in [5].

D. Coordinator

The coordinator administrates all the main operations of the search process, by giving instructions to memory and agent management modules, and receiving information from them. It is in charge of the creation of the agents at the beginning of the algorithm. The number of agents to be created is given by the parameter n_a. The locations of these agents at the start of the search process are not randomly generated, but are computed in order to maximize distances between them. The instruction to create the initial set of agents is given to the agent management module, then the MADO algorithm can start the search process. The coordinator detects also the changes in the environment. This detection is performed when all the agents have finished one loop in their search algorithm, i.e. when all the agents have moved one step ahead in their trajectory and/or performed one adaptation of their step size.

E. Parameter fitting of CMADO

Table I summarizes the nine parameters of CMADO that the user has to define. In this table, the values given are suitable for the problem at hand, and they were fixed experimentally. These values will be used to perform the experiments reported in the following section.

We fixed the number of local search agents equal to 2, because the convergence needs to be fast. Having too many agents exploring the search space will slow down the individual convergence of each agent and lead to a waste of fitness function evaluations. The precision parameter δ_p and the lowest step size r_l need to be correctly adapted to the change severity and to the change frequency of the environment. A low value for δ_p makes the agents unable to track optima in a fast changing environment, since they will spend too many iterations on fine-tuning their current solution, and might never be able to send it to the coordinator for archiving, before a new change in the environment occurs. Hence, the lower is the number of iterations between two changes, the higher should be the value of δ_p. This means that the precision of the found optima in fast changing environments might be worse than in slow changing ones. The value of r_l needs also to be well suited to the severity of the changes, since a low value of r_l in

TABLE I
CMADO PARAMETER SETTING.

Name	Value	Short description
n_a	2	initial (and average) number of local search agents (nonzero integer)
n_m	3	capacity of the archive of found optima (nonzero integer)
N	5	number of neighbor candidate solutions (nonzero integer)
c_u	0.5	weight of the cumulative dot product (real in $[0, 1]$)
c_r	0.8	step size adaptation coefficient (real in $]0, 1]$)
r_e	0.1	initial step size and exclusion radius (real in $]0, 1]$)
r_l	0.001	initial step size of tracking agents (real in $]0, 1]$)
δ_t	8	maximum number of moves an agent can make without improving its fitness more than δ_p (nonzero integer)
δ_p	1.0E-10	precision parameter of the stagnation criterion of the agents trajectory searches (positive nonzero real)

a fast changing environment requires a lot of adaptations of the step size, and a waste of many fitness function evaluations. On the opposite, a high value of r_l may make the tracking agent leave the peak of the optimum it has to track, and make it begin exploring the search space elsewhere.

IV. Results and discussion

In this section, we present the experimental results obtained using the proposed method. We first present the segmentation results using CMADO. Then, we will present the geometric matching results. We finish this section by the clinical validation of the obtained results. Our experimentations were done on 12 patients, of mean age 44.4 years, prospectively undergoing a MR examination for suspicion of a non communicating hydrocephalus. In order to illustrate the performance of the dynamic segmentation algorithm, we will present the segmentation of a sequence. As we have too many images, we show only the segmentation of the first two, and the last two images of the considered cine MRI sequence. Fig. 7(a) shows the original images: the first two and the last two images of the considered sequence, in Fig. 7(b) the corresponding original histograms are presented. The estimated histograms are presented in Fig. 7(c), and the corresponding segmentation results (in 3 classes) are in Fig. 7(d). One can remark that by merging the classes 2 and 3 (illustrated in grey and white, respectively), we can segment the lamina terminalis with a good accuracy. Fig. 8 shows the evolution of the 800 initial evaluations of the objective function, the difference between the value of the objective function of the best solution found $(fi*)$ and that of the current solution (fi) for each image (i). The index of current image segmentation is given on the axis X. The number of assessments is given on the Y axis The gap $(fi*-fi)$ is given on the axis Z. For readability, a logarithmic scale is used on the axis Z. The presented curve gives an idea about the convergence of the algorithm to an optimal value. It can also be seen as a stagnation metric of the algorithm. Moreover, the presented results are from 20 successive runs of the algorithm. The obtained results on all

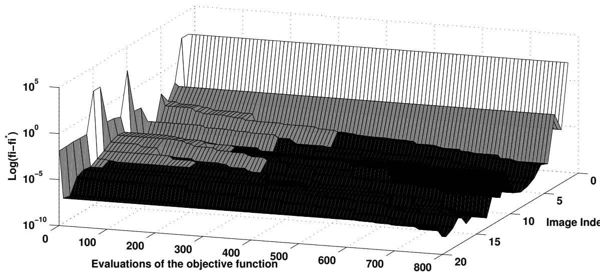


Fig. 8. Illustration of the convergence of CMADO.

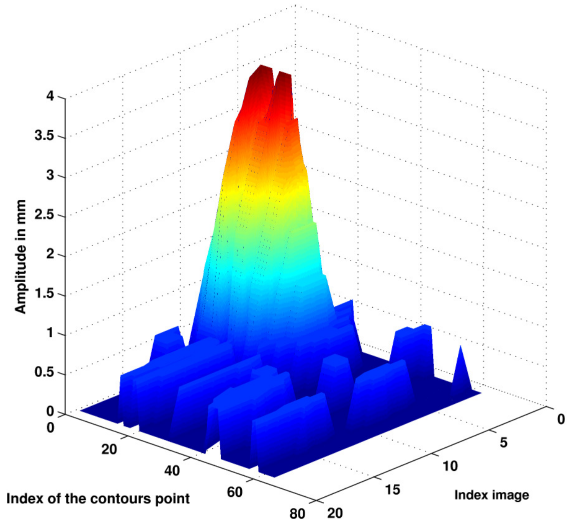


Fig. 10. Illustration of the final estimation of the amplitude of the movement of the ROI (lamina terminalis).

sequences are presented in Table II. The execution time in seconds to segment the sequence of 20 images, averaged over 20 runs is: $11.18{\mp}1.44$, and the average number of evaluations for segmenting an image sequence, averaged over 20 runs is: 7536.86 ∓ 988.30.

In order to illustrate the segmentation results, we only show one example of the segmentation (Fig. 9(b)) of the sequence presented in Fig. 9(a). In order to have a good segmentation result, we merged the classes 2 and 3. Then, we present only the final contour for all images of the sequence.

Fig. 10 presents the results after the matching procedures of successive contours on a lamina terminalis, in the third cerebral ventricle cine-MRI sequence, that consists of 20 images and encompassing the entire cardiac cycle. It illustrates the analysis of the movement of the ROI; in this example, the maximum amplitude of the movement is equal to 3.47 mm.

V. Conclusion

In this work, we show that accurate quantitative measurements of the walls of the third cerebral ventricle movements can be obtained from developed brain cine-MRI. From the wide availability of standard cine-MRI and the mostly auto-

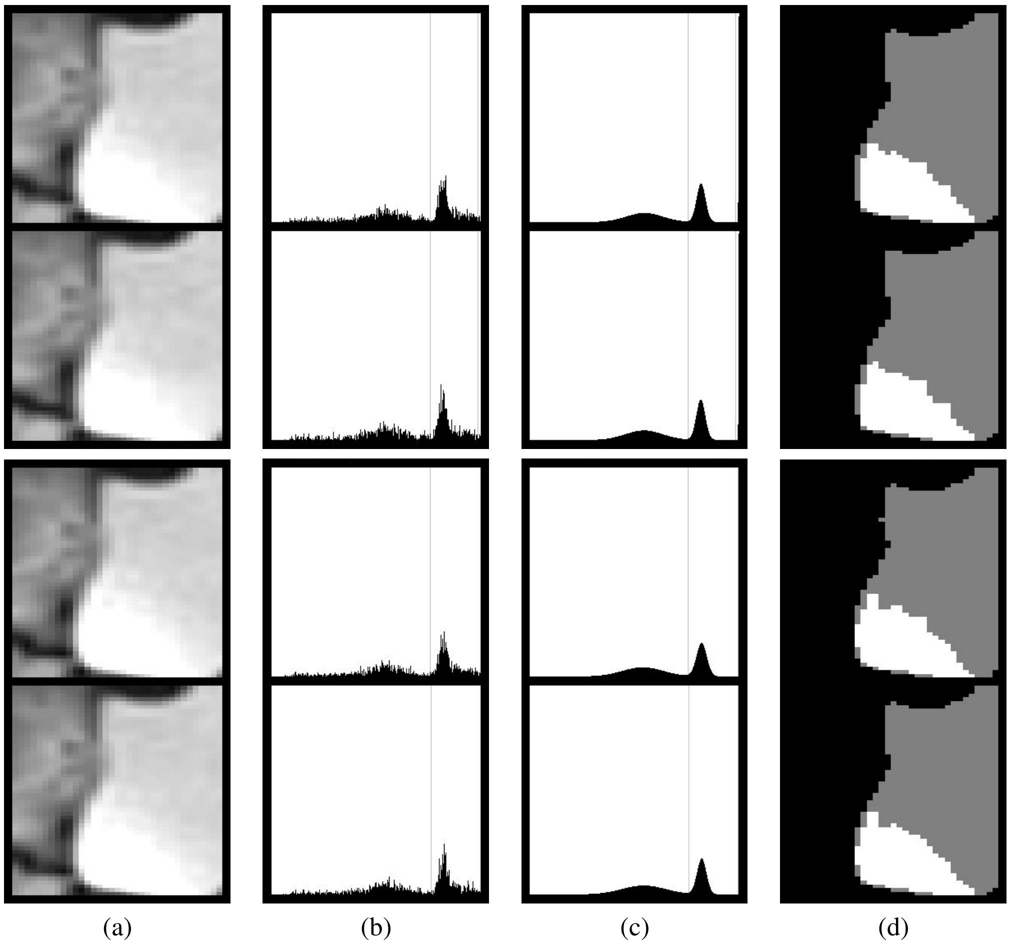


Fig. 7. Illustration of the segmentation procedure. (a) Original images, (b) Original histograms, (c) Estimated histograms, (d) Segmented images.

TABLE II

OBTAINED PARAMETERS OF THE GAUSSIANS FOR THE CONSIDERED SEQUENCE.WHERE P_i, μ_i, AND σ_i ARE THE PAREMETERS OF THE GAUSSIAN i.

Image index	μ_1	P_1	σ_1	μ_2	P_2	σ_2	μ_3	P_3	σ_3
1	140	0.0051	31.5511	209	0.0213	7.7273	255	0.1139	0.6525
2	139	0.0051	32.1886	208	0.0215	7.7592	255	0.1099	0.6206
3	137	0.0052	32.0611	209	0.0225	7.5042	255	0.1041	0.6206
4	132	0.0050	32.8261	209	0.0211	8.5242	255	0.0892	0.6206
5	131	0.0048	33.2086	209	0.0183	10.5642	255	0.0759	0.6206
6	129	0.0052	36.7167	209	0.0182	10.4491	255	0.0761	0.2252
7	131	0.0054	35.0113	210	0.0168	11.4691	255	0.0751	0.2252
8	133	0.0057	32.6526	211	0.0164	11.9791	255	0.0766	0.2252
9	136	0.0057	30.9313	211	0.0145	14.1466	255	0.0792	0.2252
10	137	0.0059	28.4451	213	0.0127	16.6009	255	0.0864	0.2252
11	139	0.0060	25.3692	214	0.0118	18.9597	255	0.0957	0.2252
12	136	0.0058	25.9748	213	0.0118	18.8641	255	0.1108	0.2252
13	134	0.0056	28.7798	212	0.0126	16.7922	255	0.1192	0.2252
14	134	0.0055	32.8120	210	0.0151	12.5209	255	0.1256	0.2252
15	134	0.0057	33.6407	209	0.0163	11.0866	255	0.1240	0.2252
16	136	0.0057	32.8757	209	0.0159	11.5647	255	0.1246	0.2252
17	137	0.0057	32.9076	209	0.0170	10.8634	255	0.1261	0.2252
18	137	0.0056	32.9395	210	0.0180	10.3534	255	0.1261	0.2252
19	138	0.0055	35.7445	210	0.0198	8.8872	255	0.1240	0.2252
20	139	0.0053	38.1670	210	0.0210	8.1222	255	0.1230	0.2252

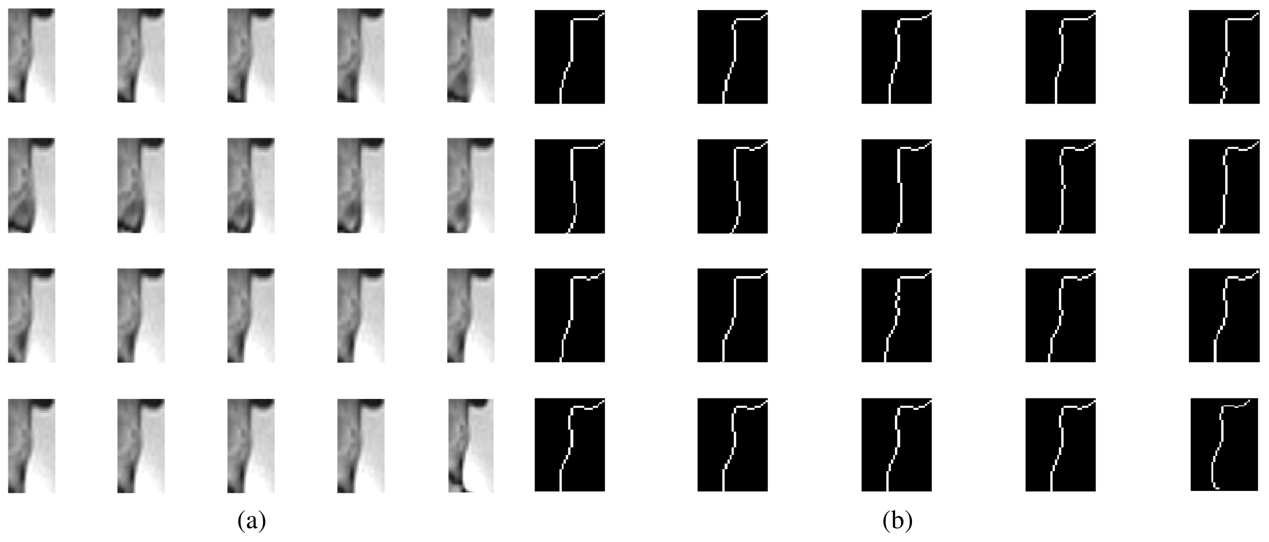


Fig. 9. Illustration of the segmentation results. (a) Original images of the selected ROI, (b) Contours of segmented images after merging classes 2 and 3.

mated analytic process, this is a new concept that has a great potential for improved clinical applicability of the quantitative assessment to the third ventricle movement. To segment quickly all the images of a sequence, a new method that takes profit from the effectiveness of the dynamic optimization paradigm is proposed. The process is sequentially applied on all the 2D images. The assessment of the ventricle deformation requires the matching of the detected contours within a cardiac cycle. Except for the initialization of the segmentation, which can be reduced to the selection of one ROI in the ventricle on the first image of the sequence, the entire procedure is fully automated and provides an accurate assessment of the ROI deformation throughout the entire cardiac cycle. Our work under progress consists in an open source software encapsuling the proposed method and the use of an elastic deformation model to register the contours.

Acknowledgment

The authors would like to thank Pr. P. Decq from *Service de Neurochirurgie, Centre Hospitalier Universitaire Henri Mondor, Unité d'Analyse et de Restauration du Mouvement (LBM ENSAM-CNRS 8005), 51 avenue du Maréchal de Lattre de Tassigny, 94000 Créteil, France*.

References

[1] J. Branke and D. Mattfeld, "Anticipation and flexibility in dynamic scheduling," *International Journal of Production Research*, vol. 43, no. 15, pp. 3103–3129, 2005.

[2] C. Rossi, M. Abderrahim, and J. C. Diaz, "Tracking moving optima using kalman-based predictions," *Evolutionary Computation*, vol. 16, no. 1, pp. 1–30, 2008.

[3] C. Rossi, A. Barrientos, and J. D. Cerro, "Two adaptive mutation operators for optima tracking in dynamic optimization problems with evolution strategies," in *Conference on Genetic and Evolutionary Computation*. London, England: ACM, 2007, pp. 697–704.

[4] A. Simoes and E. Costa, *Evolutionary Algorithms for Dynamic Environments: Prediction Using Linear Regression and Markov Chains*. Springer, 2008, pp. 306–315.

[5] J. Lepagnot, A. Nakib, H. Oulhadj, and P. Siarry, "A new multiagent algorithm for dynamic continuous optimization," *International Journal of Applied Metaheuristic Computing*, vol. 1, pp. 16–38, 2010.

[6] R. I. Lung and D. Dumitrescu, "Collaborative evolutionary swarm optimization with a Gauss chaotic sequence generator," *Innovations in Hybrid Intelligent Systems*, vol. 44, pp. 207–214, 2007.

[7] ——, "Esca: A new evolutionary-swarm cooperative algorithm," *Studies in Computational Intelligence*, vol. 129, pp. 105–114, 2008.

[8] I. Moser and T. Hendtlass, "A simple and efficient multi-component algorithm for solving dynamic function optimisation problems," in *Congress on Evolutionary Computation*. Singapore: IEEE, 2007, pp. 252–259.

[9] J. Branke, "The moving peaks benchmark," available at http://www.aifb.uni-karlsruhe.de/∼jbr/MovPeaks, 1999.

[10] J. Hodel, P. Decq, A. Rahmouni, S. Bastuji-Garin, A. Maraval, and Combes, "Brain ventricular wall movement assessed by a gated cine mr true fisp sequence in patients treated with endoscopic third ventriculostomy," *European Radiology*, vol. 19, no. 12, pp. 2789–2797, 2009.

[11] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging*, vol. 13, pp. 146–165, 2004.

[12] J. S. Weszka and R. Azriel, "Histogram modifications for threshold selection," *IEEE Trans. Syst. Man Cybernet*, vol. 9, pp. 38–52, 1979.

[13] A. Nakib, H. Oulhadj, and P. Siarry, "Non-supervised image segmentation based on multiobjective optimization," *Pattern Recognition Letters*, vol. 29, pp. 161–172, 2008.

[14] ——, "Image histogram thresholding based on multiobjective optimization," *Signal Processing*, vol. 87, pp. 2516–2534, 2007.

[15] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, 3rd ed. Springer, 1998.

[16] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.