Chapter 7

Experiments with real examples

" The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka! ' but 'That's funny ...' "

Isaac Asimov

7.1 Introduction

After the experiments carried out with synthetic data in the previous sections, this chapter presents experimental examples carried out with real data. These real problems present different characteristics and restrictions that have to be taken into account by the graph matching algorithm.

The first application is the recognition of healthy human brain MR images in three dimensions. Secondly, the inexact graph matching example of the recognition of facial features on human faces is tackled.

7.2 Recognition of brain images

7.2.1 Motivation

The recognition of internal structures of the human brain in magnetic Resonance images (MRi) using anatomical atlas presents nowadays an increasing interest, and many publications can be found in the literature on applications derived from this field in areas such as morphometry, localization of pathologies regarding abnormal structures in the brain, study of the evolution of pathologies, and support for functional studies of the brain. In all these applications the individual anatomy and the recognition of the internal structures is a preliminary step. Moreover, the step of automatically recognizing brain images by computers is a goal that could improve considerably the diagnosis and work of neurologists and radiologists.

The main difficulties of such a recognition are the differences in the orientation of the images and in gray level distributions between machines, but mainly the differences are due to the structural particularities of healthy individuals (the many variations from a person to another in composition, size and orientation on these different regions), and the presence of possible pathologies. Most of the commercial programs that nowadays perform a similar recognition procedure are based on the analysis of the grey levels of the image. Others, more sophisticated, are able of identifying regions under variation of grey level distributions on

images, although this always require the help of the physician to ensure that the detection is satisfactory enough.

More recently digital anatomical atlases have been proposed as a valid generic support for representing the different brain regions and their variability.

The use of graphs tries to encapsulate all the anatomical knowledge of a healthy brain independently of these factors. In the considered application, the atlas (or model) is represented as an attributed graph, and the images to be recognized undergo an automatic segmentation procedure after which its characteristics are also represented in the form of a data graph. The whole recognition process is therefore proposed as a graph matching problem.

Regarding the input image from which the data graph is generated, it is a common practise to apply over-segmentation to the data image previously [Perchant et al., 1999, Perchant and Bloch, 1999] to ensure that all the boundaries between regions in the model appear also in the data image. This makes graph matching easier as it ensures that a vertex in G_D is matched only against a single vertex in G_M. Unfortunately, as a consequence of this procedure even more segments are created by further subdividing the input image, which at the same time results in an increase on the number of vertices in the data graph and hence in the complexity of the problem itself. Due to this reason, the isomorphism condition is too restrictive and therefore this problem is actually regarded as an inexact graph matching one.

7.2.2 Construction of the atlas and data graphs

The atlas graph[1] that we use as a general reference has been constructed with the aid of medical doctors, and it contains a vertex for each of the brain regions such as cerebellum, ventricles, and corpus callosum with all the attributes of each of them such as approximate size, approximate position in the brain, and grey level distribution in MR images. Our particular model contains 43 regions, and therefore the model graph G_M has 43 vertices. In our model the graph is not complete and only edges between neighboring regions are considered. The model graph that we used as an example contains 336 edges.

The size of the data graph G_D used in our experiments contains 245 vertices and 1451 edges. Figure 7.1 illustrates how both the atlas and data graphs are generated.

The vertex and edge attributes that we use in our particular example have been obtained from the problem described in [Perchant, 2000, Perchant and Bloch, 2000b]. The particular attributes that are considered for both the atlas and data graphs are the grey level distribution for the vertices, and the relative position and distance for the edges. These attributes are founded on fuzzy set theory, and all the information that is required for comparing each of them is stored in the form of the necessity and possibility values. The authors also define a method to combine these fuzzy attributes to create similarity measures between vertex and edge attributes of the atlas and data graphs, and these vertex and edge similarities are the basis for creating fitness functions on their example. This method provides a means to obtain the vertex similarity value $c_N(a_D, a_M)$ between any two vertices $a_D \in V_D$ and $a_M \in V_M$, as well as the edge similarity value $c_E(e_D, e_M)$ between any two edges $e_D = (a_D^i, a_D^j) \in E_D$ and $e_M = (a_M^k, a_M^l) \in E_M$. Both $c_N(a_D, a_M)$ and $c_E(e_D, e_M)$ are normalized in $[0, 1]$.

[1]The model graph is usually known as the *atlas graph* in the literature on this particular problem.

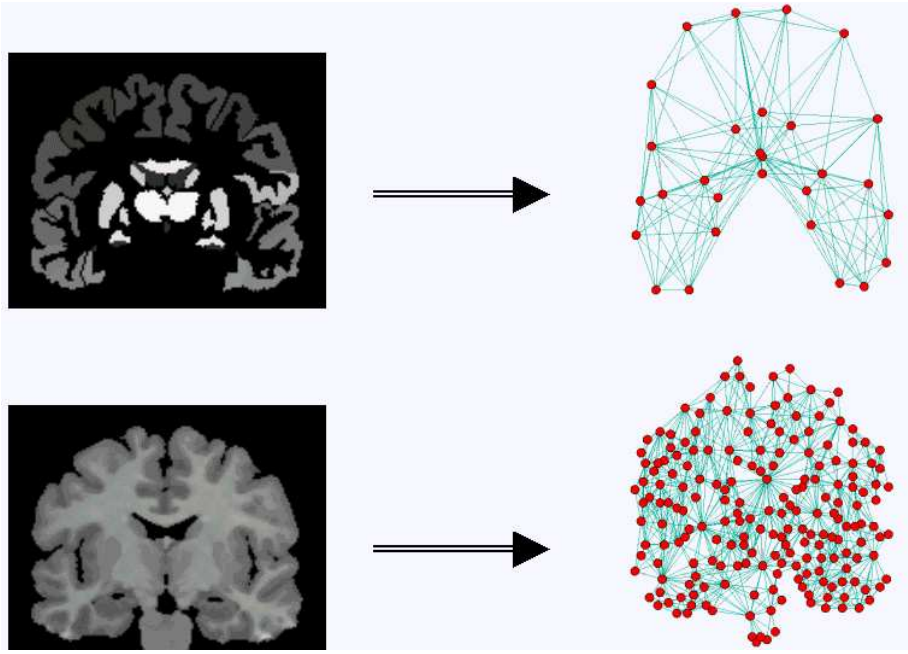

Figure 7.1: Illustration of the generation of the atlas (above) and the data graph (below). The atlas is created from a real 3D MR image of a healthy human brain manually segmented as we can see above. The data graph G_D is created segmenting the input image to be recognized. All the images are always in 3D, although here we show only a coronal view of them. Both graphs on the right show the complexity of the problem. The graph matching procedure has to assign a model vertex for each of the vertices in the data graph.

7.2.3 Description of the problem and the graph matching approach

Figure 7.2 shows the matching procedure of this problem and the meaning of the results. This figure illustrates how the successful recognition of the caudate nucleus (one of the brain regions) is performed using graph matching. As the image is acquired in 3D, two columns corresponding to an axial and a sagittal view are presented. The over-segmentation is also evident and can be appreciated in the data graph compared to the atlas graph. In the input image, the caudate nucleus is represented by two different segments in this particular example, although depending on the segmentation process this also could have been divided in even more segments (or may be in just a single one).

In the atlas (above) one vertex represents the caudate nucleus, and edges from this vertex to the neighboring vertices represent spatial relationships between them. The images below show the corresponding parts identified as the caudate nucleus once the atlas and data graph have been matched satisfactorily.

Any successful graph matching method is expected to obtain as a final result the matching illustrated in the row below for the case of the caudate nucleus. Similar examples could be given for the rest of the brain segments. However, it is important to point out that the search process relies on the fitness function and the individual representation, and therefore the proper choice of these two aspects will be responsible to a large extent for obtaining a satisfactory result at the end. The fitness function that we have selected for our experiments with this example has been obtained from the literature in [Perchant et al., 1999] and it

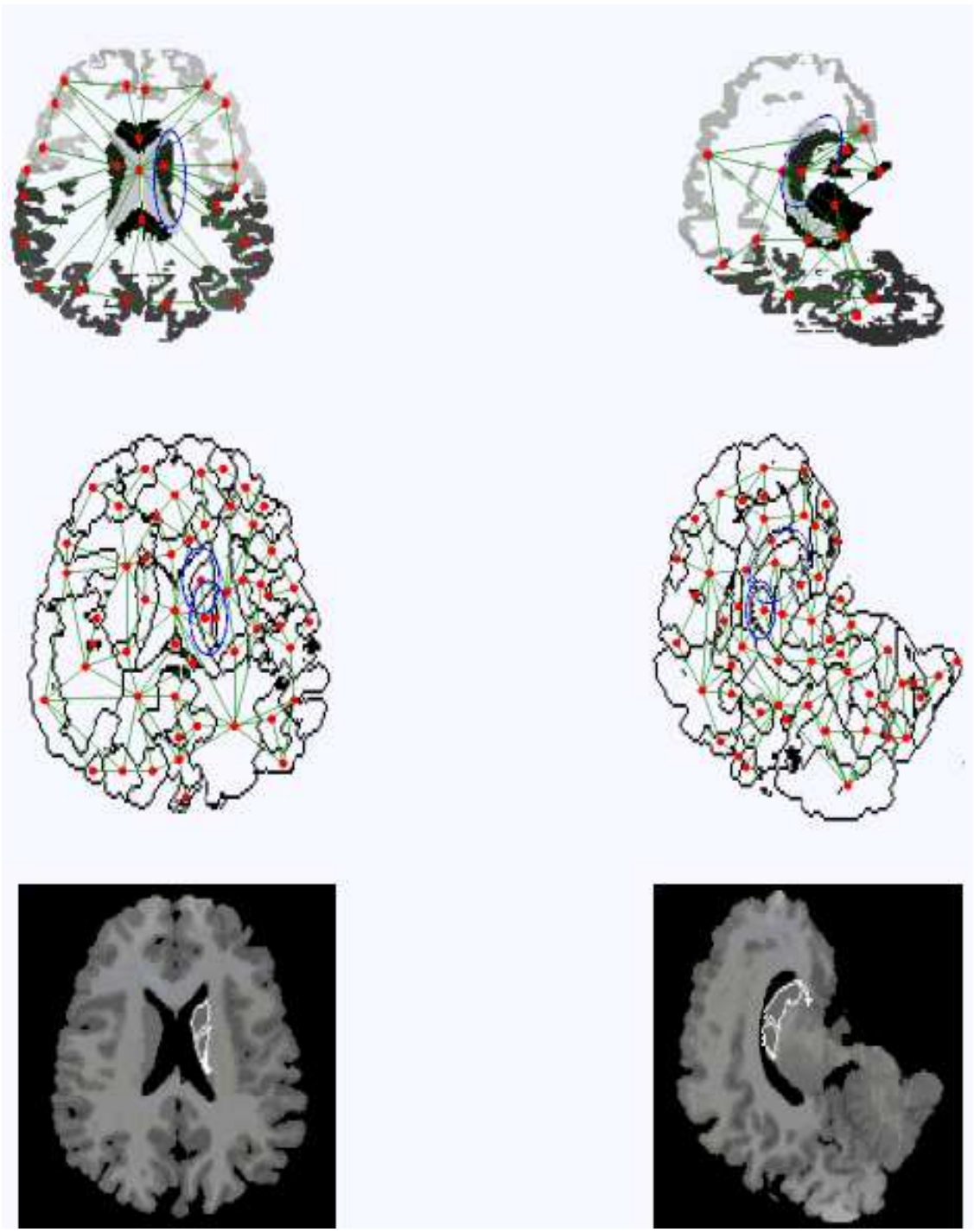

Figure 7.2: Example of the graph matching process applied to the recognition of structures in MR human brain images. In each row we have an axial and a sagittal view of the model graph, data graph, and a result respectively. All these images concentrate on the recognition of a particular segment of the brain among all the ones to be identified: the caudate nucleus.

follows the idea of the one introduced in Equation 3.2 in Section 3.4.3. On the other hand, the individual representations used here for applying discrete and continuous EDAs are the ones described in Sections 3.3.1 and 3.3.2 respectively. The procedure to evaluate individuals in the continuous domain with the same fitness function is the one explained also in Section 3.3.2 and in Appendix A. In both the discrete and continuous domains, $\alpha = 0.4$ was considered following the conclusions of [Perchant et al., 1999].

This real problem has also specific constraints that have to be satisfied for any solution in order to accept it as valid. The conditions of this problem are exactly the same as in the synthetic case analyzed in Section 6.2: no dummy vertices are allowed, only a matching is possible for each vertex in the data graph, and there is an additional constraint so that there must exist at least a region in the input image assigned to every vertex in the model. The latter condition is justified since we are dealing only with healthy brains and therefore all the brain regions have to be identified. All these aspects have already been discussed and formalized in Section 3.3.3.

7.2.4 Experimental results

Description of the experiment

In this section we compare the performances of EDA algorithms to each other and to a broadly known GA, the GENITOR [Whitley and Kauth, 1988] –a steady state (ssGA) type algorithm. Both EDAs and GENITOR are implemented in ANSI C++ language, and the experiment was executed on a two processor Ultra 80 Sun computer under Solaris version 7 with 1 GByte of RAM.

The initial population for all the algorithms was created using the same random generation procedure based on a uniform distribution. In the discrete case, all the algorithms were set to finish the search when a maximum of 100 generations or when uniformity in the population was reached. GENITOR, because of being a ssGA algorithm, only generates an individual at each iteration, but it was programmed in order to generate the same number of individuals as in discrete EDAs by allowing more iterations (201900 individuals). In the continuous case, the ending criterion was to reach 301850 evaluations (i.e. number of different individuals generated).

In EDAs, the following parameters were used: a population of 2000 individuals ($R = 2000$), from which a subset of the best 1000 are selected ($N = 1000$) to estimate the probability, and the elitist approach was chosen (that is, always the best individual is included for the next population and 1999 individuals are simulated). In GENITOR a population of 2000 individuals was also set, with a mutation rate of $p_m = \frac{1}{|V_D|}$ and a crossover probability of $p_c = 1$. The crossover and mutation operators used in GENITOR are CX [Oliver et al., 1987] and EM [Banzhaf, 1990] respectively.

Results

Results such as the best individual obtained, the CPU time, and the number of evaluations to reach the final solution were recorded for each of the experiments. The CPU time is given as a measure to illustrate the different computation complexity of all the algorithms.

Each algorithm was executed 10 times. The non-parametric tests of Kruskal-Wallis [Kruskal and Wallis, 1952] and Mann-Whitney [Mann and Whitney, 1947] were used to

	Best fitness value	Execution time	Number of evaluations
UMDA	0.7186	00:53:29	85958
MIMIC	0.7027	00:57:30	83179
EBNA_{BIC}	0.7167	01:50:39	85958
UMDA_c	0.7450	03:01:05	301850
MIMIC_c	0.7480	03:01:07	301850
EGNA_{BGe}	0.7469	04:13:39	301850
ssGA	0.6936	07:31:26	201900
	$p < 0.001$	$p < 0.001$	$p < 0.001$

Table 7.1: Mean values of experimental results after 10 executions for each algorithm for the inexact graph matching problem of the human brain structure recognition.

test the null hypothesis of the same distribution densities for all –or some– of them[2]. This task was done with the statistical package S.P.S.S. release 9.00. The results for the tests applied to all the algorithms are shown in Table 7.1. The study of particular algorithms gives the following results:

- Between algorithms of similar complexity only:
  - UMDA vs. UMDA_c. Fitness value: $p < 0.001$; CPU time: $p < 0.001$; Evaluations: $p < 0.001$.
  - MIMIC vs. MIMIC_c. Fitness value: $p < 0.001$; CPU time: $p < 0.001$; Evaluations: $p < 0.001$.
  - EBNA vs. EGNA. Fitness value: $p < 0.001$; CPU time: $p < 0.001$; Evaluations: $p < 0.001$.

  These results show that the differences between EDAs in the discrete and continuous domains are significant in all the cases analyzed, meaning that the behavior of selecting a discrete learning algorithm or its equivalent in the continuous domain leads to very different results. It is important to note that the number of evaluations was expected to be different, as the ending criteria for the discrete and continuous domains are also different. In all the cases, continuous EDAs obtained a fitter individual, but the CPU time and number of individuals created was also bigger.

- Between discrete algorithms only:
  - Fitness value: $p < 0.001$. CPU time: $p < 0.001$. Evaluations: $p < 0.001$.

  In this case statistically significant different results are also obtained in fitness value, CPU time, and number of evaluations. The discrete algorithm that obtained the best result was UMDA, closely followed by EBNA. The differences in the CPU time are also according to the complexity of the learning algorithm they apply. Finally, the results show that MIMIC required significantly less individuals to converge (to reach the uniformity in the population), whereas the other two EDA algorithms require nearly the same number of evaluations to converge. The genetic algorithm GENITOR is far behind the performance of EDAs. The computation time is also a factor to

[2]The interested reader is referred to [Siegel, 1956] for further explanations on non-parametric tests.

consider: the fact that GENITOR requires about 7 hours for each execution shows the complexity of the graph matching problem.

- Between continuous algorithms only:

  – Fitness value: $p = 0.342$. CPU time: $p < 0.001$. Evaluations: $p = 1.000$.

  Differences in fitness values between all the continuous EDAs appear to be not significant. As expected, the CPU time required for each of them is according to the complexity of the learning algorithm. On the other hand, the fact of having the same number of evaluations is due to the same ending criterion. Speaking about the differences in computation time between discrete and continuous EDA algorithms, it is important to note that the latter ones require all the 300000 individuals to be generated before they finish the search. The computation time for the continuous algorithms is also longer than their discrete equivalents as a result of several factors: firstly, due to the higher number of evaluations they perform each execution, secondly because of the longer individual-to-solution translation procedure that has to be done for each of the individuals generated[3], and lastly, as a result of the longer time required to learn the model in continuous spaces.

We can conclude from these results that generally speaking continuous algorithms perform better than discrete ones, either when comparing all of them in general or only with algorithms of equivalent complexity.

7.2.5 Computational complexity and parallelization of the problem

The complexity of the human brain recognition problem is high due to the inherent complexity of the brain structures and the fact of having 3D images [Bengoetxea et al., 2002b]. Unfortunately, such a high complexity results in very long computation time if a satisfactory matching is to be found. This section focuses on this aspect to demonstrate the need and usefulness of applying parallelization techniques.

Analysis of the execution times for the most important parts of the sequential EDA program

Section 5.5.2 underlines the need of analyzing the potential for parallelization of a program by profiling it. That section showed how to perform this step for the specific case of EBNA$_{BIC}$ using the GNU gprof tool. Table 7.2 summarizes the most salient results for this particular problem.

The BIC function computes the BIC score –i.e. the goodness of the probabilistic structure to represent the data. This function is executed many times each generation, until the best Bayesian network is found. This score constitutes essentially the whole learning of the Bayesian network. Table 7.2 shows clearly that in EBNA$_{BIC}$ the most time consuming function is BIC, the one that is used to evaluate the different Bayesian networks for representing the interdependencies between the variables.

We exploit the use of the two parallel versions of the EBNA$_{BIC}$ algorithm for this problem, the first based on threads and shared memory, and the second on processes and message passing.

[3]See Section 3.3.2 for a further explanation of this individual-to-solution procedure.

EDA type	Internal function	Relative CPU time
$EBNA_{BIC}$	BIC (Learn probabilistic model)	85.7 %
	Evaluation of individuals (Fitness function)	12.3 %
	Simulation	1.4 %

Table 7.2: Time of computing for the human brain recognition graph matching problem solved with $EBNA_{BIC}$. All the figures in the last column are given in relative times, i.e. 100 % = full execution time.

	sequential	parallel 2 workers	parallel 4 workers	parallel 6 workers	parallel 8 workers
Machine 1	26:49:48	17:40	16:55	17:08	16:11
Machine 2	10:54:15	7:14	6:56	6:50	6:57

Table 7.3: Execution times for the human brain recognition problem using the $EBNA_{BIC}$ algorithm regarding its sequential and shared memory based parallel *pthread* version for computing the BIC score (hh:mm). Results show important improvements in execution times.

Experimental results using threads and shared memory

The parallel version of the BIC function using threads does not compute any different algorithm regarding the sequential BIC function. For this reason, the best fitness values obtained with the parallel version of the $EBNA_{BIC}$ approach are exactly the same obtained with the sequential program. The only difference is just the execution time required. Two small SMP machines with different hardware configurations have been used to evaluate the performance of a parallel pthreads-based version of the $EBNA_{BIC}$ program, which characteristics are as follows: the first computer, *Machine 1*, contains 2 Intel Pentium II processors at 350 MHz, 512 KB cache, and 128 MB of RAM; the second computer, *Machine 2*, has 2 Intel Pentium III processors at 1 GHz, 256 KB cache, and 512 MB of RAM. Both machines use the operating system GNU-Linux and its programming environment. Table 7.3 compares execution times for sequential and parallel versions of $EBNA_{BIC}$. Note that parallelism does not come for free: the parallel master-worker paradigm can be applied to a single worker –thus, achieving no parallelism at all. Using Machine 2 with a single-worker parallel program, the execution time is 12:08:49, i.e., more than 1 hour slower than the original, sequential version. This cost comes from spawning the worker thread and synchronizing it with the master.

Of course, real improvement can only be apparent when two or more worker threads run in parallel. Table 7.3 shows that, with two workers runs are much shorter. We tried using more than two worker threads: additional gains are achieved, but not very significant. Generally speaking, the results obtained could be summarized as follows: parallel programming techniques based on shared memory appear to be an effective acceleration tool for very CPU-consuming programs such as EDAs. Programmers need to familiarize themselves with libraries such as *pthreads* (or equivalent ones) to take full advantage of the potential of parallel machines.

Experimental results using message passing

A similar experiment was performed using the parallel version of the $EBNA_{BIC}$ algorithm based on MPI. The program was executed in the cluster which characteristics were explained

	parallel 2 workers	parallel 4 workers	parallel 6 workers	parallel 8 workers	parallel 10 workers
Fast Ethernet	27:04	19:21	17:23	15:59	17:25
Myrinet	17:36	13:04	11:41	10:53	10:42

Table 7.4: Execution times for the human brain recognition problem using the EBNA$_{BIC}$ algorithm regarding its sequential and message passing based parallel MPI version for computing the BIC score (hh:mm). The execution time of the sequential version on one of the machines is 26 hours and 49 minutes. Results show the validity of the parallel system.

in Section 6.4.2. As well as in that section, the Fast Ethernet and Myrinet computer networks were tested. The results are shown in Table 7.4. The results obtained also show considerable reductions in the computation time, specially for the Myrinet case. Similar results as in the previous section are obtained, although the performance of MPI is still better than such of the pthreads one.

7.3 Recognition of human facial features

7.3.1 Motivation

Face analysis and recognition have received intense and growing attention from the computer vision community, partially because of the many applications such as human-computer interaction, model-based coding, teleconferencing, security and surveillance. A particularly important task that arises in different problems of face recognition is the location and segmentation of *facial feature regions*, such as eyebrows, iris, lips, and nostrils [Pantic and Rothkrantz, 2000]. Some of the most popular methods such as techniques based on template matching, integral projections, and so on are also discussed in [Pantic and Rothkrantz, 2000].

We propose the use of graph matching to solve this problem. In this approach, we have again one graph representing a model of a face and an image for which recognition has to be performed. Both the model and data graphs are built from regions and relationships between regions, and there are vertex and edge attributes in both of them too. The idea is to model basic knowledge about the features in a face as a graph. Based on an over-segmentation, image information is also represented as a graph. More specifically, an attributed relational graph is used to represent each image. Therefore, the recognition procedure expects to find a suitable matching between both graphs. The introduced technique can be applied to both static images and to video sequences.

7.3.2 Construction of the model and data graphs

The model graph

The easiest way of obtaining the model graph $G_M = (V_M, E_M)$ is by manually segmenting a face image selected specifically for this purpose. The attributes of the model graph are later computed regarding data extracted from that image. Figure 7.3 illustrates the way of generating the model graph from a selected image. Firstly, the facial landmarks are located by a tracking method [Feris and Cesar, 2001] and used to define a limited region. The model is manually obtained from a reference image. Figure 7.3c shows the face image that has been manually segmented to define the face model, and that has been used in the preliminary

experiments reported in this section. A more robust approach is to segment a set of face images and to calculate the graph attributes from different images. This approach allows the inclusion of vertices corresponding to specific facial features that may not be present in the selected model image, such as teeth or beard. The associated vertex and edge attributes are calculated only from the images where they are effectively present. Complete graphs are used, and even edges from and to the same vertex are considered. As a result, if $|V_M| = n$ then $\forall a_M^i \in V_M$ there are n edges in E_M so that $(a_M^i, a_M^j) \in E_M$ with $j = 1, 2, \ldots, n$.

The model graph is generated so that it contains vertices associated to each facial feature of interest, e.g. for each eyebrow, iris, nostril, mouth and the skin. The proposed approach allows additionally inclusion of other facial features non always included such as glasses, teeth and beard.

It is important to note that in the model of Figure 7.3c, some single facial features have been subdivided deliberately. An example of this are the eyebrows, which are subdivided in three parts each. This has been done because the adopted vertex and edge attributes are calculated based on average measures over the segmented image regions. Therefore, model attributes extracted from large regions tend to be less representative because such regions often present larger variability with respect to the attributes. That is why some facial features have been subdivided in order to circumvent such a potential problem. In addition, the fact that the skin is not a well-localized facial feature (in contrary to pupils and nostrils) presents an additional difficulty for introducing structural relations between skin and the other features. As an example, while it is possible to define structural relations such as *the pupil is above the nostrils*, it would be more difficult to define a similar relation with respect to the skin. As a solution to this problem, alternative types of structural relations such as *surrounded by* are avoided. Instead, we have adopted the approach of further dividing the skin into sub-segments as shown in Figure 7.3c, which allows us to use the same relational attributes between all facial features. An additional reason for over-segmenting the input image is that –as well as in the previous example of the human brain– this procedure will ensure that each of the segments will correspond to only one model region.

It is worth emphasizing that subdividing the model regions implies increasing the number of graph vertices and edges and thus it increases the complexity of the whole problem. Therefore, there is a trade-off between quality of model attributes and computation time that should be carefully considered when designing the graph model.

Generation of the data graph

The initial step to segment the facial feature regions is to locate the face in the image, which can be done both in photograph images and in video sequences. The latter requires detection of the face in a frame and to track it in the subsequent frames. In approaches such as [Cesar and Bloch, 2001, Cesar et al., 2002a], these location and tracking steps are performed using the Gabor Wavelet Network (GWN) [Kruger and Sommer, 1999]. The GWN acts as a rigid model that provides *approximative landmarks* which are located near the facial features to be segmented [Feris and Cesar, 2001] (e.g. eyes, nose and mouth). These landmarks are used in two different ways in order to make our approach more efficient: firstly, only certain regions around the landmarks are considered, and secondly, the landmark information is used by the optimization algorithm to constrain the solution search space. The reader is referred to [Feris and Cesar, 2001] for further details on the GWN approach. We will call *super-regions* to the 4 landmarks that we consider on our particular example, which are centered each on the left pupil, right pupil, the nose, and the mouth.

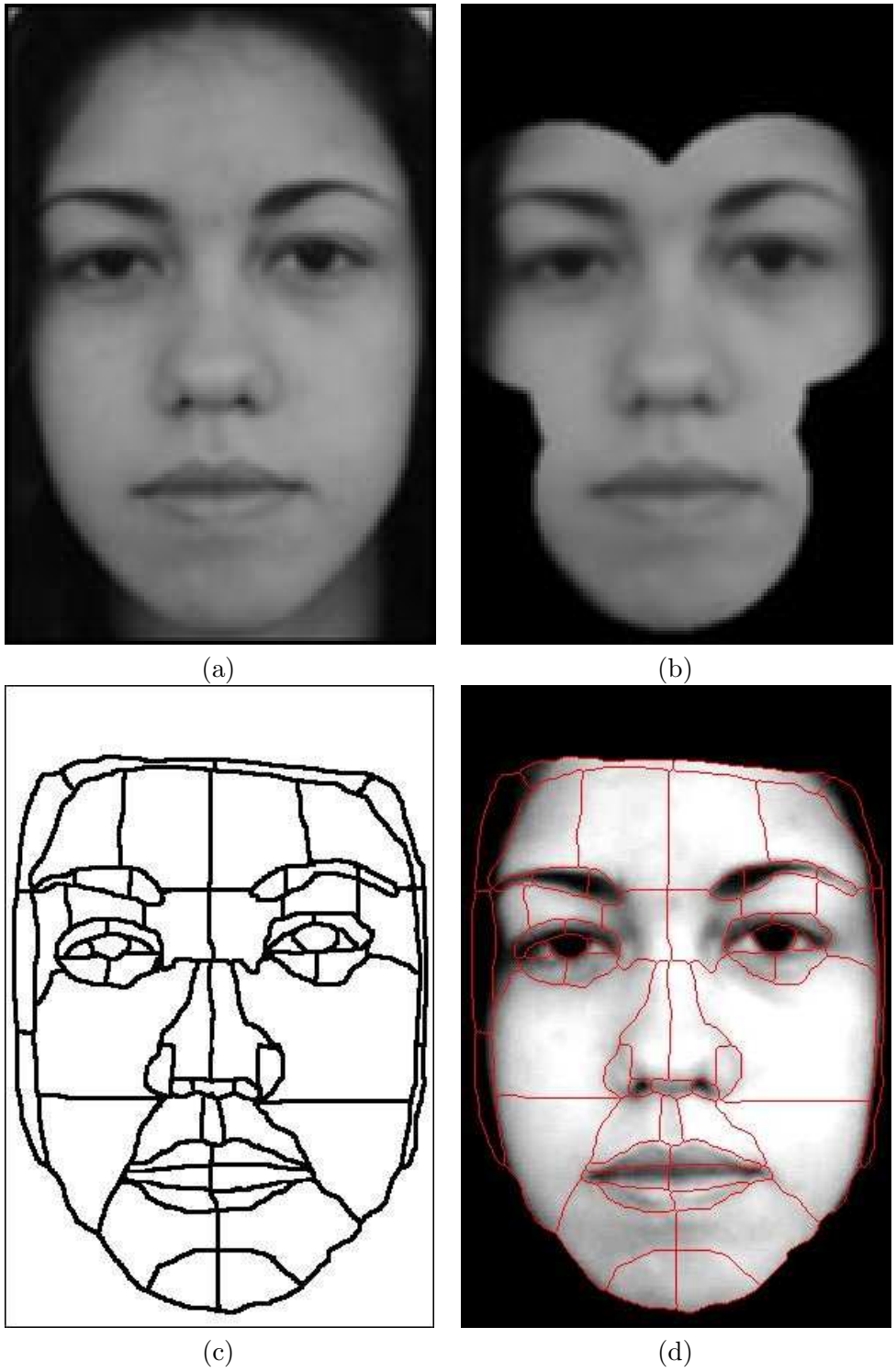

Figure 7.3: Illustration of the process to generate the model graph: (a) an original image is selected to extract the model from; (b) a masked version which contains only the regions of interest around the landmarks is obtained; (c) the face model is obtained by manual segmentation; (d) image where the model is superimposed to the face image (just for explanatory purposes).

Figure 7.4: Example of an over-segmented image after applying the watershed algorithm.

The input image is over-segmented using a watershed algorithm based on the approximative landmarks of the facial feature regions (the four super-regions in our case), which results in an over-segmented image such as the one shown in Figure 7.4. The over-segmentation is performed since it is important that the edges that define the facial regions to be segmented are also present after the segmentation process –similarly as in the brain recognition problem. The resulting over-segmented image is the one that is used next to generate the attributed data graph G_D, which is later matched against a model graph through a graph matching algorithm. As with the model graph, each vertex of G_D is adjacent to all other G_D vertices, and it is also adjacent to itself. Under these conditions no bijective correspondence can be expected because of the over-segmented nature of the image regarding the model, and as a result this problem calls for inexact graph matching.

Vertex and edge attributes and similarities

The knowledge about face features is represented in the attributed graphs by the unary (vertex) and binary (edge) attributes of both the model and data graphs. Unary attributes are calculated from the segmented region of the corresponding vertex, while relational attributes are based on the spatial disposition of the regions. We have based our decision on which type of attributes to use regarding the work described on this field such as [Cesar and Bloch, 2001, Cesar et al., 2002a,b], where a detailed description of vertex and edge attributes for this particular graph matching problem can be found. Following these references, the attributes and their similarities where selected as follows:

Unary attributes: Let $G = (V, E)$ be the graph (either the model or data graph), then $\eta(a) = (g(a), w(a), g_{min}(a), l(a))$ with $a \in V$ is the unary attribute of the region cor-

responding to vertex a in the segmented image, where $g(a)$ denotes the average grey level, $w(a)$ is computed as a texture index from wavelet coefficients, $g_{min}(a)$ denotes the average grey-level of the 15% darkest pixels of the image region associated to vertex a, and $l(a)$ is a super-region number assigned by the tracking procedure regarding approximative landmarks as described in the previous section. Both $g(a)$ and $g_{min}(a)$ are normalized between 0 and 1 with respect to the minimum and maximum possible grey-levels, and the value of 15% used to calculate $g_{min}(a)$ is a parameter that may be changed depending on the nature of the input image.

Regarding the definition of the vertex similarity, the vertex attributes $g_{min}(a)$ and $l(a)$ are included for supporting the recognition process, but only the vertex attributes $g(a)$ and $w(a)$ are considered in the computation of the similarity between vertices: $g_{min}(a)$ is included in order to facilitate the recognition of textured regions composed of light and dark pixels (e.g. the eyebrows and in some cases the mouth), and the super-region label $l(a)$ provided by the tracking procedure is only taken into account to restrict the search space. As a result, the similarity between any two vertices $a_D \in V_D$ and $a_M \in V_M$, denoted by $c_N(a_D, a_M)$, is defined as:

$$c_N(a_D, a_M) = 1 - (\ \beta\ |g(a_D) - g(a_M)|\ +\ (1-\beta)|w(a_D) - w(a_M)|\ )$$

where $0 \leq \beta \leq 1$ is a parameter for tuning the relative importance between the vertex attributes, and $c_N(a_D, a_M)$ is normalized to $[0, 1]$ with a higher value representing a higher similarity between the two vertices.

Binary attributes: Let $a^i, a^j \in V$ be any two vertices of G. The relational attribute $\nu(a^i, a^j)$ of the edge $(a^i, a^j) \in E$ is defined as the vector $\overrightarrow{v}_{a^i,a^j} = \frac{\overrightarrow{p_{a^i} p_{a^j}}}{2d_{max}}$, where d_{max} is the largest distance between any two points of the masked face regions, and p_{a^i} and p_{a^j} are the centers of gravity of the respective image regions. Following this particular definition of binary attributes, we have clearly that, $\nu(a^i, a^j) = -\nu(a^j, a^i)$, and as a result edges are directed. Note also that we allow $a^i = a^j$, and therefore the edge $(a^i, a^i) \in E$ is also considered, as well as its attribute $\nu(a^i, a^i)$. The latter are considered as a reference for comparing edges of regions $a1_D, a2_D \in V_D$ when the homomorphism h satisfies that $h(a1_D) = h(a2_D) = a1_M$ (in that case we would compare the edges $(a1_D, a2_D)$ and $(a1_M, a1_M)$). Note that this comparison is not considered in all the fitness functions proposed in Section 3.4.1.

Analogously as for vertex similarity, the similarity between any two edges $e_D = (a^i_D, a^j_D) \in E_D$ and $e_M = (a^k_M, a^l_M) \in E_M$ is defined as:

$$c_E(e_D, e_M) = 1 - \left| \overrightarrow{v}(a^i_D, a^j_D) - \overrightarrow{v}(a^k_M, a^l_M) \right|$$

where $\overrightarrow{v}(a, a')$ is the vector that goes from the center of gravity of region a to such of region a'. Again, we will only consider for a global similarity value those $c_E(e_D, e_M)$ that satisfy the condition $a^k_M = h(a^i_D), a^l_M = h(a^j_D) \in V_M$. This similarity is also normalized to $[0, 1]$.

7.3.3 Description of two face feature recognition problems and the graph matching approach

This section illustrates the facial feature recognition problem using a first reduced example (only taking into account the segments of an eye) and a second one representing a whole

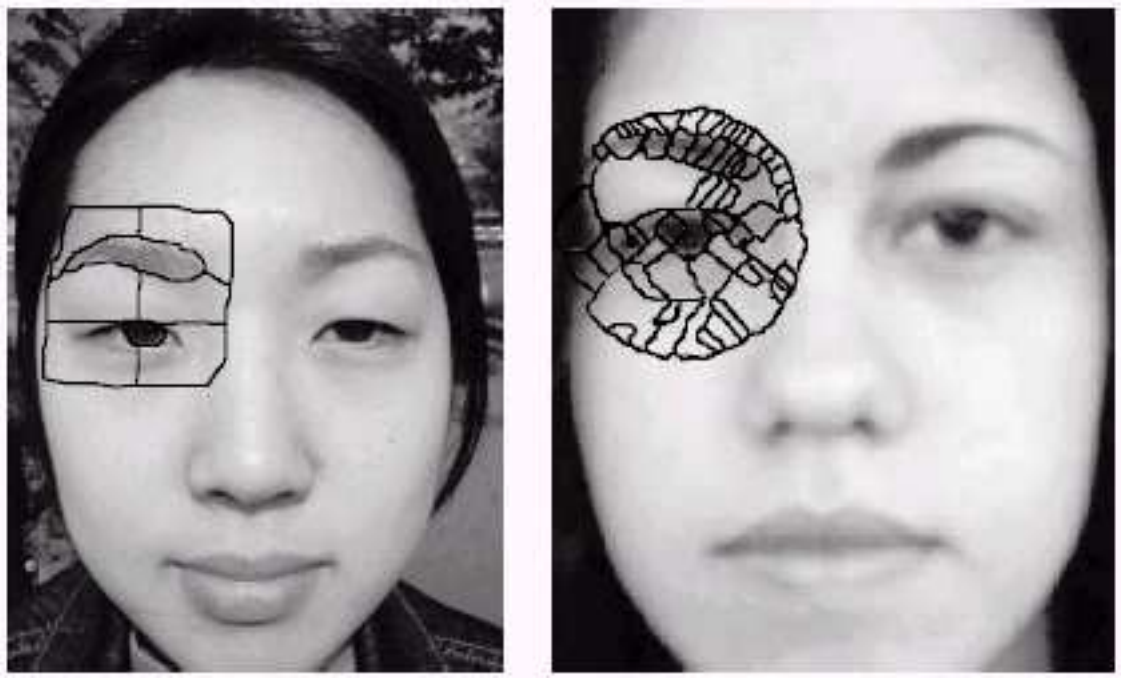

Figure 7.5: Illustration of the first (reduced) example of the facial feature recognition problem. These images show the small parts selected from the images to create the model graph G_M (left) and the data graph (right).

human face. In these examples, both the model and data graphs are attributed, and the recognition will rely on similarity functions between attributes. We use in both examples the model and data graph generation techniques described in the previous section.

The best homomorphism is found by optimizing a fitness function based on object and relational attributes defined on the graphs. Discrete EDAs and GAs are applied in both examples.

Problem 1: a reduced example for only an eye

This problem is proposed as a combinatorial optimization one. The fitness function that we have selected for our experiments is the $f_1(h)$ one described in Section 3.4.2, with $\alpha = 0.4$ and $\beta = 0.7$. On the other hand, only discrete EDAs will be applied and the individual representations used here is the one described in Section 3.3.1. This problem does not contain any specific constraints such as the ones required in the human brain recognition problem, although in the final solution no dummy vertices are allowed and only a matching is possible for each vertex in the data graph. Figure 7.5 illustrates the whole problem by showing the model and data images. In this example, the model contains 13 regions and the data image 75 ($|V_M| = 13$ and $|V_D| = 75$).

Problem 2: the whole face example

This second example analyzes other aspects in whole face examples. The model used for this case is again the one shown in Figure 7.3, which has been already applied in the literature [Cesar et al., 2002a,b] and was obtained using a standard digital camera. Input face

Algorithm	$\min_h(f_1(h))$	# of errors
UMDA	0.9455	5 (6.6%)
MIMIC	0.9455	5 (6.6%)
EBNA_{BIC}	0.9455	5 (6.6%)
EBNA_{K2}	0.9455	5 (6.6%)
eGA	0.9094	54 (71.0%)
ssGA	0.9454	7 (9.3%)

Table 7.5: Summary of the results for the small facial feature recognition problem.

images to which recognition is to be performed have been obtained using the same camera, but some images have also been downloaded from standard public databases available on-line[4]. The input or target images have been selected in order to show the robustness of the method regarding images acquired in different conditions (i.e. geometry, illumination, distance from the camera, etc.).

Four different face images were analyzed, and Table 7.6 shows for each of them the number of segments and edges after the automatic over-segmentation procedure. The model used is shown in Figure 7.3 and it contains 62 vertices and 3844 edges.

7.3.4 Experimental results

Problem 1

The results obtained with the reduced example of the eye regions for all the algorithms are summarized in Table 7.5. The number of incorrectly labelled regions is computed based on an optimum solution drawn by labelling a manually segmented data image. All EDAs provide the same solution. Although their computation time is higher, they lead to a better optimization compared to GAs. The 5 errors that are present in the returned solution are also included in the solutions provided by the other algorithms. The ssGA and specially the eGA approaches performed somewhat worse.

Figure 7.6 illustrates the same result obtained for all the EDAs. The most important features to be identified in the surrounding region of only one eye are the pupil, and the eyebrow. This figure shows these relevant parts following the final graph matching solution obtained with EDAs. The segments encircled in white are the ones matched satisfactorily as pupil or eyebrow. The ones encircled in red correspond to the 5 matching errors. Among these 5 errors, 3 correspond to very tiny regions, for which even the manual result given as the optimal cannot be considered reliable. Another error corresponds to a region in the hair, and exhibits one of the limits of the proposed approach, in cases where the data image contains objects that are not represented in the model. An extension of this approach could be to allow for no recognition in such cases (i.e allowing the use of dummy vertices). Finally, the last error corresponds to a region at one extremity of the eyebrow, which is labelled as skin. The reason is that the contour between the eyebrow and the skin was not detected by the watershed algorithm and therefore this region contains also some skin.

The skin is divided in the model deliberately in several regions in order to avoid geo-

[4]Examples of face images can be found for instance on the web site at the University of Bern, available at `http://iamwww.unibe.ch/~fkiwww/staff/achermann.html` and from the University of Stirling, available at `http://pics.psych.stir.ac.uk/`.

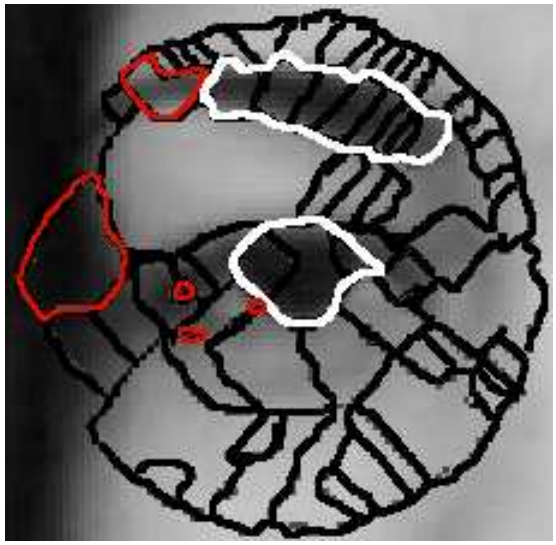

Figure 7.6: Illustration of the solution obtained with the different EDAs for the first (reduced) example of the facial feature recognition problem. The regions encircled in white correspond to the segments matched as pupil and eyebrow satisfactorily, while the ones in red represent the errors in the solution.

	deise	f014	f041	m036
vertices	112	176	183	228
edges	12544	30976	33489	51984

Table 7.6: Figures of the 4 cases that are analyzed in the second example, illustrating the number of vertices and edges that are considered. These values are illustrated for showing the difference in complexity for each of the examples.

metrical properties such as *surrounded by*. These skin section appear in the final solution to be correctly recognized. This satisfactory result illustrates the influence of edge attributes when vertex attributes are too similar for differentiation of regions.

Problem 2

We present firstly some results obtained applying only the $f_1(h)$ fitness function described in Section 3.4.2. These results are shown in Figures 7.7 to 7.10. Each of these figures shows the obtained segmentation and recognition of the eyebrows, nostrils and lips using the following search algorithms: (a) UMDA, (b) MIMIC, (c) EBNA$_{BIC}$, and (d) ssGA.

The results illustrated in Figures 7.7 to 7.10 are discussed in detail next. We executed 5 times each of the algorithms for each of the examples. Table 7.7 shows the results in the form of the mean fitness value of the best individual at the last generation, the number of different individuals created during the search, and the CPU time. The latter computation time is presented as a measure to note the difference in computation complexity of all the algorithms. The machine in which all the executions were performed is a two processor Ultra 80 Sun computer under Solaris version 7 with 1 Gb of RAM.

The null hypothesis of the same distribution densities was tested (non-parametric tests of Kruskal-Wallis and Mann-Whitney) for each of the examples and algorithm executions with

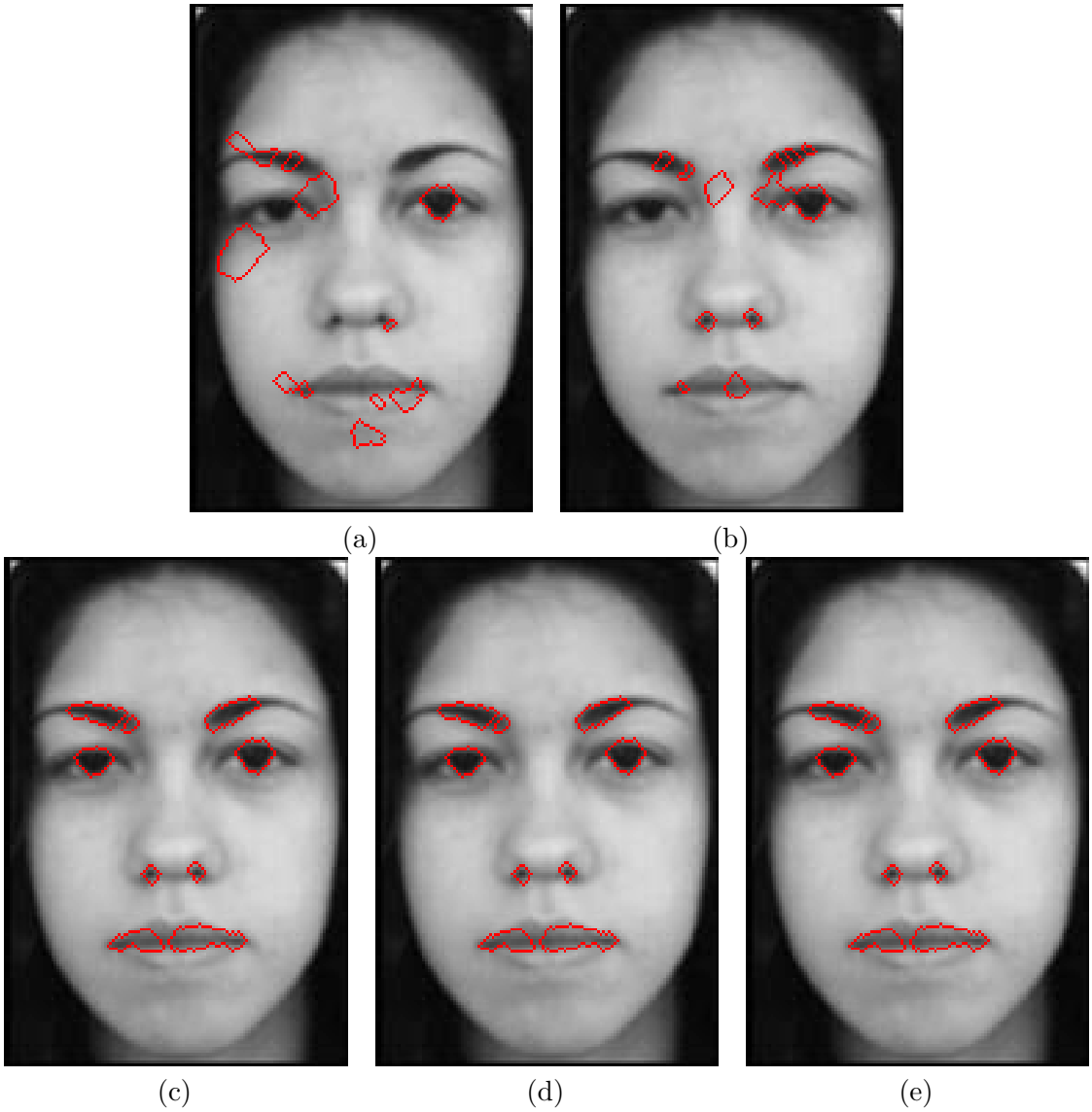

Figure 7.7: Segmentation and recognition of facial features for the *deise* example using (a) eGA, (b) ssGA, (c) UMDA, (d) MIMIC and (e) EBNA$_{BIC}$. As the model has been extracted from this image, the target and the model image are the same for this case.

the statistical package S.P.S.S. release 10.1. The results of these tests are shown in Table 7.8, and they confirm the significance of the differences of all the algorithms regarding the value of the best solution obtained of EDAs and GAs. They also show that differences between the different EDAs in the best individual obtained are not statistically significant, but these are significant among eGA and ssGA. In all the examples the EDAs obtained better results than the GAs, and these differences appear to be statistically significant regarding Table 7.8. Also, the differences in execution time appear to be significant in all the algorithms, and EDAs required in all cases more time. As a result, regarding Tables 7.7 and 7.8, we can conclude that the results are much better in EDAs at the expense of a higher computation time, but EDAs arrive to a more satisfactory final individual by having to evaluate less

Figure 7.8: Segmentation and recognition of facial features for the example *f014* using (a) eGA, (b) ssGA, (c) UMDA, (d) MIMIC and (e) EBNA$_{BIC}$.

individuals than GAs. This fact is important to take into account if the computation of the fitness function is more complex (i.e. if it requires more CPU time) than the one selected for our experiments.

Table 7.9 shows the errors obtained for some images in the form of a ground-truth-based assessment (total number of misclassified regions and percentage with respect to the total number of regions in the segmented image) for the different optimization algorithms. As it can also be seen, the ssGA and eGA algorithms lead to much poorer results with respect to the recognized facial features and the best solution obtained. These results could be partially understood by the fact that these two GAs are both general purpose algorithms. The use of GAs specially thought for our problem could lead to better results[5]. However, it must also be said that the EDAs applied are also general purpose ones.

In the light of the results obtained for the fitness values, we can sum them all up as follows: generally speaking, EDAs obtained in all the executions a fitter individual than GAs, but

[5]After the many years that genetic algorithms are part of the literature, very different versions and adaptations have been proposed. Therefore, other versions of GAs or even a different choice of crossover and mutation operators could lead to a better behavior.

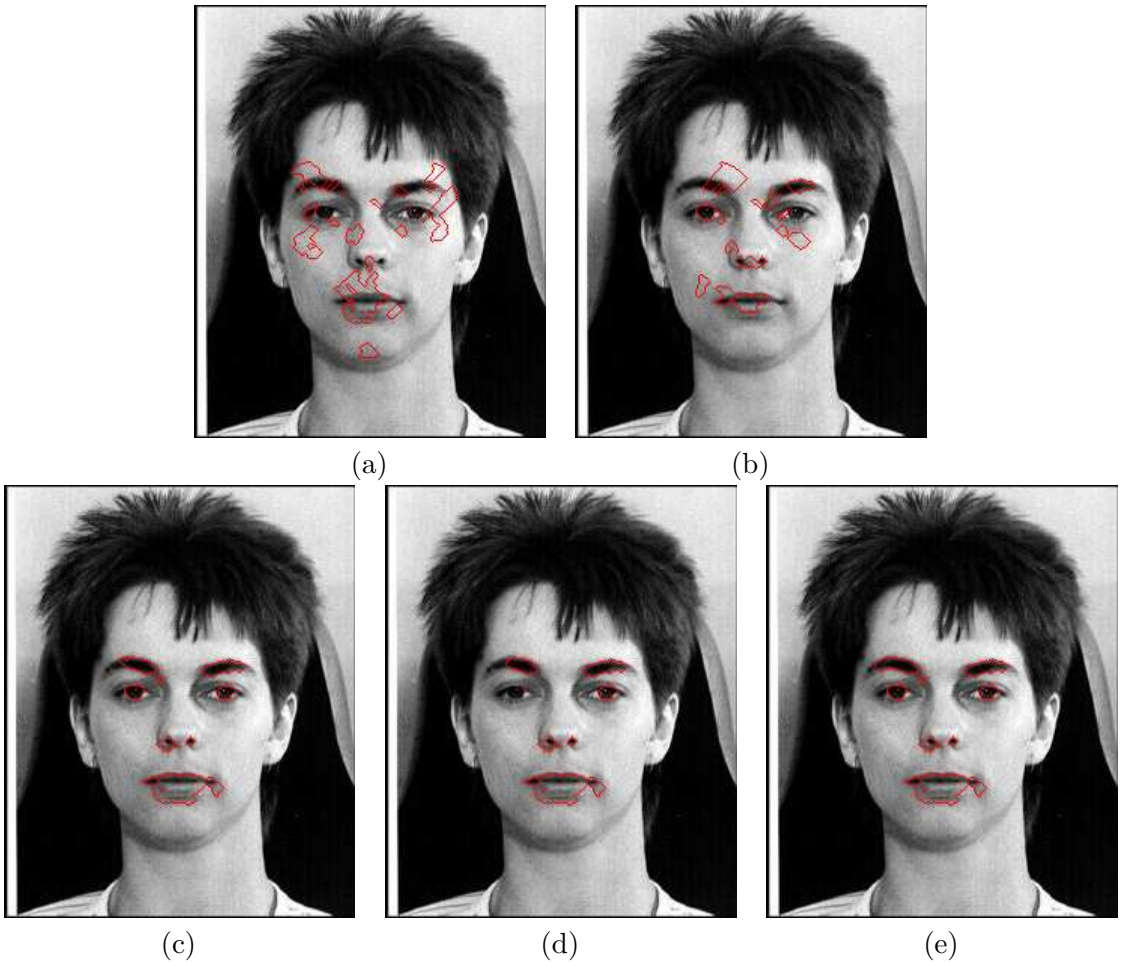

Figure 7.9: Segmentation and recognition of facial features for example *f041* using (a) eGA, (b) ssGA, (b) UMDA, (c) MIMIC and (d) EBNA$_{BIC}$.

although the number of individuals created is lower, the CPU time required was bigger. Besides the important problem of assessing the optimization algorithms with respect to the criterion function and execution time, it is also important to analyze the obtained results with respect to the problem context, i.e. recognition of the facial features of interest. In order to perform this task, we have generated a ground-truth for some faces by manually labelling the over-segmented images, i.e. by obtaining a human solution for the problem. Of course, such ground-truth is prone to some subjectivity, since a human operator decides the label of each region of the over-segmented image with respect to the model (nearby regions around the facial features are generally difficult to be classified, being often labelled differently depending on the operator). Then, the ground-truth is compared to each automatically labelled image to obtain the number of errors, i.e. number of misclassified regions (with respect to the ground-truth). Some of the error regions are shown in Figure 7.11. This figure shows 3 types of errors found in our experiments:

- Very small regions nearby the facial features (indicated by A in Figure 7.11), which have generally been missed by the operator while producing the ground-truth.

Figure 7.10: Segmentation and recognition of facial features for the example *m036* using (a) ssGA, (b) UMDA, (c) MIMIC and (d) EBNA$_{BIC}$.

- Regions in the outer portion of the eyebrows (indicated by B in Figure 7.11), which have been left out during the classification procedure because of the above explained reasons.
- True errors such as matching to a wrong region (indicated by C in Figure 7.11).

As it can be seen, the proposed method is able to correctly recognize the facial features of interest, being robust to substantial differences between the model (Figure 7.3d) and the target image. A problem that has been identified in our experiments is that the outer portions of the eyebrows in the model contain several skin pixels, leading to misclassifications near it. Therefore, only the two inner portions that compose each eyebrow in the model have been identified as *eyebrows* by the graph matching algorithms. Because of the structural constraints in the fitness function $f_1(h)$, the outer portions of the eyebrows in the obtained results have not been included, which is a drawback that we intend to circumvent in future work.

It is interesting to note that the facial features that have been recognized could be used for matching the model over the input face, being suitable for updating the model parameters with respect to the target image. The latter would be useful for instance to apply this technique for face and facial feature tracking in video sequences, thus avoiding the need for

	deise			f014		
	best	Time	eval.	best	time	eval.
UMDA	0.6896	00:13:54	143909	0.6827	00:58:26	184894
MIMIC	0.6894	00:16:53	129008	0.6827	01:07:46	167117
$EBNA_{BIC}$	0.6898	01:19:20	153924	0.6830	04:40:16	185908
eGA	0.5656	00:33:00	202000	0.5357	01:30:35	202000
ssGA	0.6429	00:25:54	202000	0.6120	01:07:50	202000
	f041			m036		
	best	time	eval.	best	time	eval.
UMDA	0.6813	00:58:09	196702	0.6794	01:41:22	201900
MIMIC	0.6813	01:09:51	181910	0.6790	02:08:22	201900
$EBNA_{BIC}$	0.6813	05:15:19	1951	0.6794	09:11:07	201900
eGA	0.5374	01:35:02	202000	–	–	–
ssGA	0.6107	01:08:58	202000	0.598	01:53:35	202000

Table 7.7: Figures of the 4 cases that we analyzed, illustrating the mean values after 5 executions of each of the algorithms. The *best* column corresponds to the mean best fitness value obtained through the search, and the differences between the algorithms are evident as EDAs obtain the best results for all the examples. The *time* column shows the CPU time required for the search, and the *eval.* one shows the number of individuals that had to be evaluated in order to end the search.

		GAs-EDAs	among GAs	among EDAs
deise	best	$p < 0.001$	$p = 0.008$	$p = 0.078$
	eval.	$p < 0.001$	$p = 1.000$	$p = 0.068$
	time	$p = 0.121$	$p = 0.008$	$p < 0.001$
f014	best	$p < 0.001$	$p = 0.008$	$p = 0.105$
	eval.	$p < 0.001$	$p = 1.000$	$p = 0.064$
	time	$p = 0.495$	$p = 0.008$	$p = 0.002$
f041	best	$p < 0.001$	$p = 0.008$	$p = 0.811$
	eval.	$p < 0.001$	$p = 1.000$	$p = 0.012$
	time	$p = 0.643$	$p = 0.008$	$p < 0.002$
m036	best	$p < 0.001$	–	$p = 0.085$
	eval.	$p < 0.001$	–	$p = 1.000$
	time	$p = 0.306$	–	$p = 0.002$

Table 7.8: Statistical significance for all the 4 examples and algorithms after 5 executions of each of the algorithms, by means of the results of the non-parametric tests of Kruskal-Wallis and Mann-Whitney. The first column shows the result of the test comparing all EDAs with all GAs, the second is the test for comparing eGA and ssGA, and the third shows the comparison between the three EDAs.

the GWN detection and tracking every sequence [Cesar et al., 2002b, Costa and Cesar, 2001].

7.4 Conclusions of the experiments on real problems

Different approaches for brain image recognition and for facial feature segmentation based on graph homomorphisms have been proposed. In this context, we have defined graph-based model representations of a human brain and a face model respectively, as well as the proce-

	deise		f014		f041		m036	
	Errors	%	Errors	%	Errors	%	Errors	%
UMDA	1	0.89	12	6.82	7	3.83	12	5.26
MIMIC	1	0.89	12	6.82	8	4.37	15	6.58
$EBNA_{BIC}$	1	0.89	11	6.25	5	2.73	15	6.58
eGA	21	18.75	36	20.45	46	25.14	50	21.93
ssGA	35	31.25	63	35.80	57	31.15	-	-

Table 7.9: Table showing the number of misclassified regions in each test image for each algorithm. The *Errors* column indicates the number of misclassified regions, while the column *%* shows the percentage with respect to the total number of regions in the image.

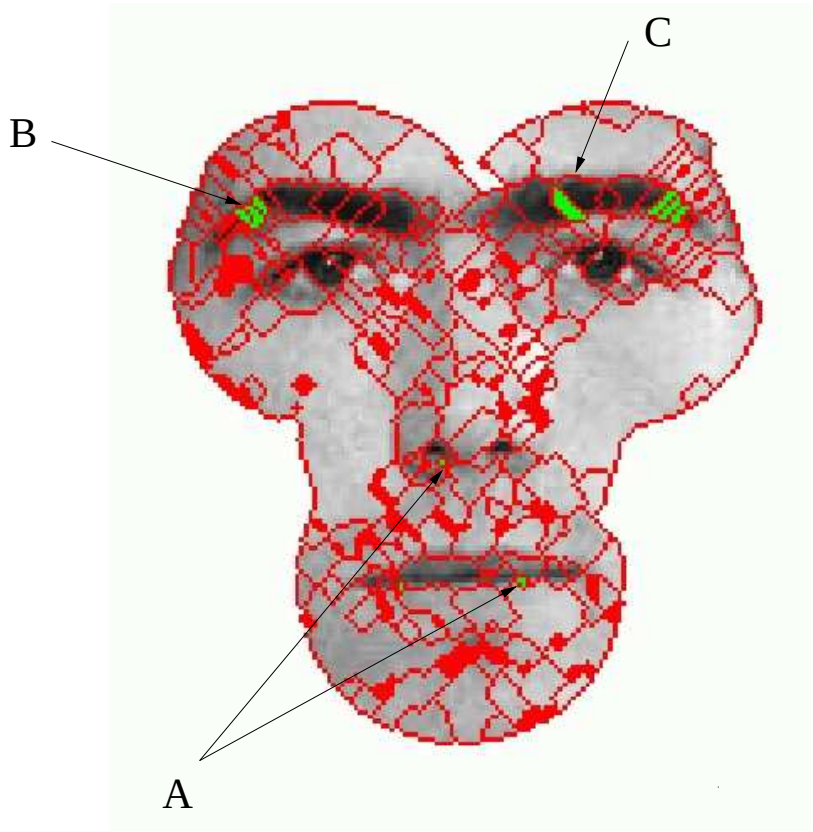

Figure 7.11: Example of some typical error regions. A: two very small regions nearby facial features, which are too small to be properly identified. B: regions in the outer portion of the eyebrows that could be ambiguous to classify as eyebrow or skin, in this case recognized as skin. C: true matching error, in this case eyebrow region recognized as part of the pupil.

dures for automatic segmentation of input images for these real problems. Fitness functions and individual representations for these problems have been tested with GAs and EDAs. The image processing techniques for the facial feature recognition problem are more sophisticated, since the facial features are segmented taking advantage of tracking information provided by a GWN technique. Our ongoing work aims at improving the method robustness and at generalizing it in a number of different ways, e.g. using fuzzy morphisms [Perchant

and Bloch, 2002], developing other object and relational attributes, and taking advantage of the homomorphism found in the previous frame in a video sequence when searching for the one in the current frame. These image processing techniques (except the temporal aspect) remain also to be tested for the human brain recognition problem.

Furthermore, the above definition of graph homomorphism implies that all vertices in G_D are mapped to G_M and if the input face presents features not known by the model, they will be classified. In the case of the facial feature recognition problem, if the input face has glasses and the model graph does not include them, the glass regions will be classified as skin or some other facial feature. Two possible solutions to this problem can be designed. The first one is to leave some of the G_D vertices unmapped, thus relaxing the homomorphism approach. The second approach is to define a *dummy vertex* in the model graph, to which the unclassified input regions should be mapped. Both approaches present specific difficulties and are currently being considered in our research. Also, other fitness functions are currently under investigation. Finally, we aim at applying our graph-based face model to face and facial expression recognition problems. In particular, we would like to analyze if temporal changes in the relational attributes could be used for facial expression recognition, because such trajectories are important for perceiving changes in facial expressions.