TASK QUARTERLY 16 No 1, 145–162

A PARALLEL GENETIC ALGORITHM FOR CREATING VIRTUAL PORTRAITS OF HISTORICAL FIGURES

HENRYK KRAWCZYK, JERZY PROFICZ
AND TOMASZ ZIÓŁKOWSKI

Faculty of Electronics, Telecommunications and Informatics,
Gdansk University of Technology,
Narutowicza 11/12, 80-233 Gdansk, Poland
hkrawk@eti.pg.gda.pl, jerp@task.gda.pl, tziol@eti.pg.gda.pl

(Received 10 June 2012)

Abstract: In this paper we present a genetic algorithm (GA) for creating hypothetical virtual portraits of historical figures and other individuals whose facial appearance is unknown. Our algorithm uses existing portraits of random people from a specific historical period and social background to evolve a set of face images potentially resembling the person whose image is to be found. We then use portraits of the person's relatives to judge which of the evolved images are most likely to resemble his/her actual appearance. Unlike typical GAs, our algorithm uses a new supervised form of *fitness function* which itself is affected by the evolution process. Additional description of requested facial features can be provided to further influence the final solution (*i.e.* the virtual portrait). We present an example of a virtual portrait created by our algorithm. Finally, the performance of a parallel implementation developed for the KASKADA platform is presented and evaluated.

Keywords: genetic algorithms, fitness function, KASKADA platform, parallel processing, high-performance computing

1. Introduction

Throughout history, there lived a great number of extraordinary people who managed to leave their mark on the world and whom we thus remember to this day. Alas, due to the technology limits of the past and historical or personal circumstances, their physical appearance often remains unknown to us. It has always been the domain of artists to, through their imagination and skill, recreate the possible appearance of such figures. But then an intriguing question was raised: could we conduct a similar process using modern high-performance computing technology and artificial intelligence algorithms?

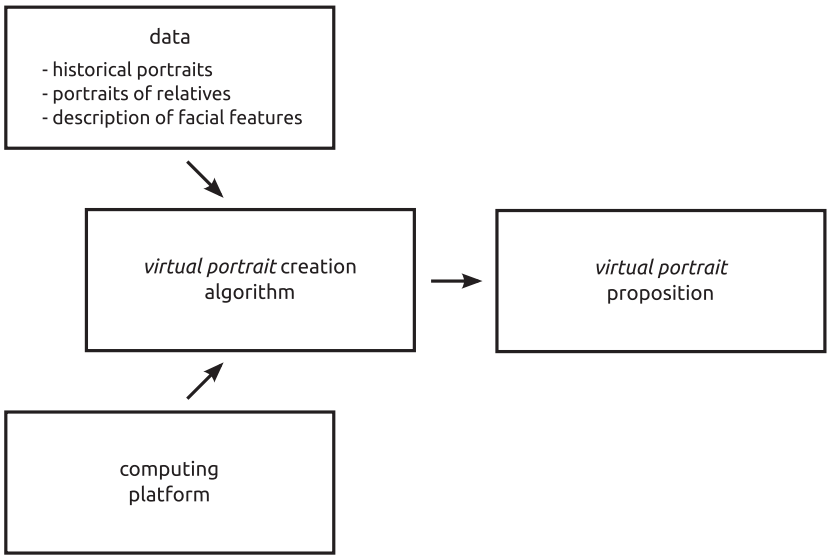

Figure 1. Virtual portrait creation – problem overview

We propose a genetic algorithm which can be used to create such a *virtual portrait* of a historical figure whose exact appearance is unknown, but whose family members' portraits are available. It is of course impossible for us to create a portrait perfectly faithful to reality, as there exists no scientific method which would allow us to perform such a reconstruction. But considering the fact that there is often significant resemblance among relatives, it may be possible to create a virtual image which would at least give us some idea about what the historical figure *could* have looked like. The general idea is presented in Figure 1.

1.1. Adopted concepts and methods

Genetic algorithms (GAs) are a popular artificial intelligence method for solving optimisation problems through mimicking the process of evolution. There is a *population* of potential solutions which are subject to *recombination/reproduction* (producing new solutions from existing ones) and *mutation* (introducing small random modifications to the solutions). The solutions are rated by a *fitness function* which corresponds to the function whose optimal value is to be found. As in natural selection, there is *selection pressure* which favours the most fit solutions (*i.e.* those which are closest to the optimum). There are different methods of performing the selection process, *e.g. roulette-wheel selection* or *tournament selection*. For an introductory description of GAs, see [1].

Variations of genetic algorithms have been proposed for dealing with complex optimisation problems, such as *multiobjective genetic algorithms (MOGAs)* [2–6]. It has also been suggested that it can prove beneficial to tweak the formula of the fitness function during consecutive iterations [7].

For analysing similarities between different face images, we use the *eigenfaces* algorithm. The algorithm extracts characteristic facial features from images

by comparing them with an 'average' face computed from a set of various portraits of random people. These extracted features (eigenfaces) can then be used for determining similarity between provided face images. For a detailed description of the eigenfaces method, see [8, 9].

Face image transformations are realised in our algorithm through a technique called *morphing*, often used in computer generated animations. Morphing uses a pair of 2D images or 3D models and through mathematical transformations computes a new one which resembles both of the originals in the pair (see [10–12]). In our case, the *thin plate spline* (see [13]) mathematical model is used internally for *2D image warping*.

The image analysis and transformation processes performed by our algorithm require high-performance computing techniques. Fortunately, the recombination, mutation and rating phases of GAs can generally be conducted independently for different elements of the population, and therefore such algorithms can usually benefit from parallel processing techniques (see [14, 15]).

1.2. Our contribution

The main idea of our genetic algorithm is to use portraits of people living in the appropriate historical period to 'evolve' a set of new faces which would potentially be reminiscent of the actual appearance of the historical figure. The process of 'evolution' is controlled through comparison with available portraits of the historical figure's relatives and a set of provided descriptions of facial features.

One of the main problems associated with developing the algorithm was the fact that a clear definition of an appropriate fitness function is difficult to formulate. Furthermore, the set of input images on which the function is based tends to be very limited in size (only a limited set of original historical portraits may be available).

As a novel contribution introduced in our algorithm, we propose a new form of a fitness function whose values are affected by the genetic evolution process. This means that as newly generated face images are added to the population during every iteration, they are used to update the set of images used by the fitness function and so improve the process of fitness rating. Consequently, the values of the fitness function change together with the evolution of the population. We argue that this technique allows us to obtain better final results as well as overcome the problem of limited size of the input portraits set. Furthermore, our function can also be 'supervised' through additional parameters, which allow supplementary information about facial features to be incorporated into the fitness rating process. This supervision provides additional control over the appearance of the created virtual portraits.

To achieve acceptable computation times, we developed a parallel implementation of our algorithm. For this purpose, we used the KASKADA platform (see [16, 17]), which allowed us to efficiently develop and execute the implementation on the *Galera+* compute cluster [18].

In the next section, we provide a detailed description of our algorithm and various parameters which can be used to control it. We then present an example case of creating a virtual portrait with our algorithm. In the final section, a parallel implementation for the KASKADA platform is presented together with a performance evaluation.

2. Algorithm description

As mentioned in the Introduction, our algorithm is in many ways analogous to classical genetic algorithms [1]. There is a population of face images, which are rated, selected, recombined and mutated to form new *generations*. A general overview of the algorithm is presented in Figure 2.

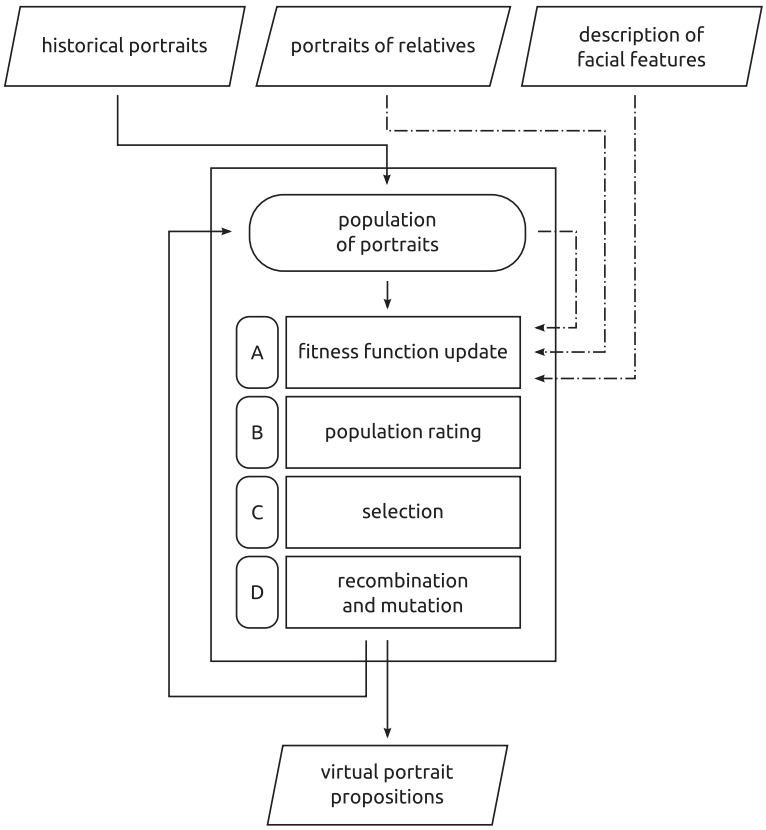

Figure 2. An overview of the algorithm

2.1. Input data

The input data set consists of portraits of random people living in the same period and having a similar social background as the historical figure whose virtual portrait we want to create. Each face image is additionally described by locations of characteristic points marking specific facial features (*e.g.* eyes or mouth, see

Figure 3. Historical portraits with facial features marked through characteristic points

Figure 3). The points are determined manually using a visual software tool and are stored in XML documents. This set of images is inserted into the initial population and participates in the evolution.

By using historical portraits as a base for our algorithm we want to assure that the generated virtual portraits would have a general appearance appropriate for the specific historical period.

2.2. Fitness function update and population rating

Similarly to other GAs, our algorithm uses a fitness function to rate the evolved specimens so that the best ('fittest') results can be identified. An overview of the proposed dynamic fitness function is presented in Figure 4. As can be seen in the illustration, the rating process is based on different data sets.

Firstly, there is a set of portraits of relatives of the historical figure. Every portrait has a certain *weight* associated with it, which depends on the degree of consanguinity, *e.g.* a father's portrait would have a higher weight than a cousin's portrait.

As described in the Introduction, an eigenface-based face recognition algorithm is used by the fitness function for rating the evolved faces. This approach requires a base set of face images which is used to generate a set of eigenfaces. For this purpose, we use two image sets: the original historical portraits and the virtual portraits generated in the previous iterations. After this step, any face image can be compared with the obtained set of eigenfaces. For each portrait, a vector is returned which describes the portrait's similarity to the faces from the base set (the vector defines the portrait's position in an n-dimensional space, where n is the number of the eigenfaces). This vector serves as a numerical description of the associated portrait. The difference between two such vectors serves as a measure of similarity between the two corresponding face images. Thus, the eigenfaces algorithm allows us to compare the appearance of the generated virtual portraits

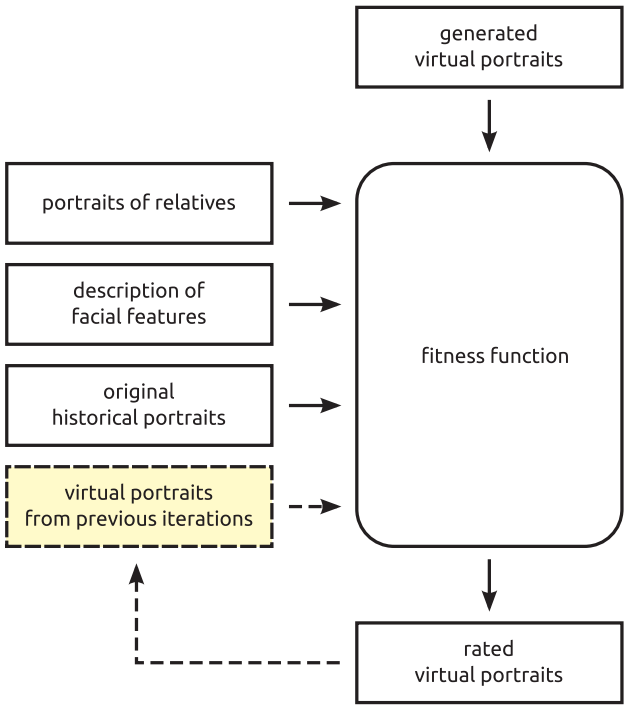

Figure 4. An overview of the proposed dynamic fitness function

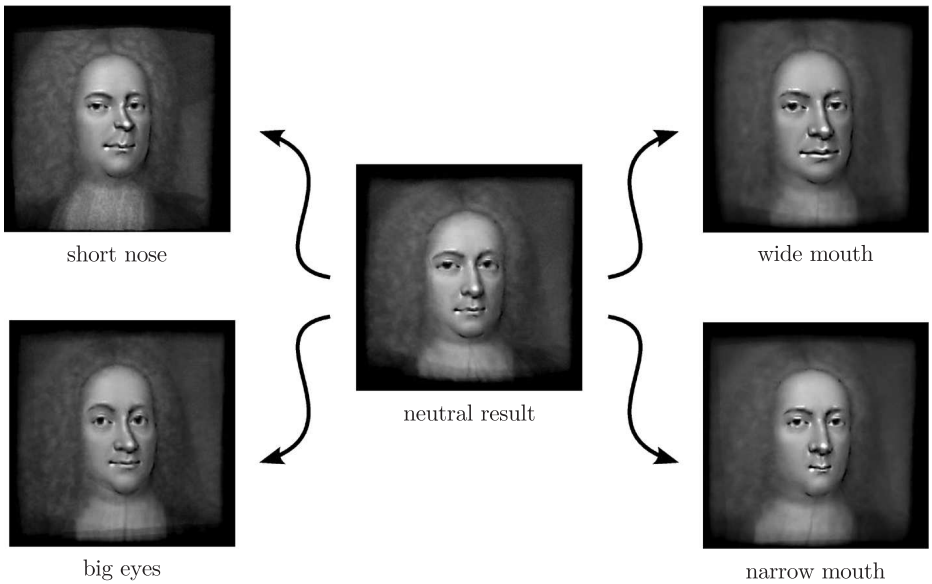

Figure 5. Examples of different results achieved through providing different sets of requested facial features

to the provided set of relatives' images. The portraits with a higher degree of similarity are assigned a better fitness rating.

Our algorithm allows additional information about facial appearance to be incorporated into the fitness function by providing a description of facial features. The preferred shape of individual face parts (*e.g.* eyes size, nose shape, mouth width, *etc.*) can be specified by providing locations of appropriate characteristic points. These points can be, for example, established from a textual description in a way analogous to what was presented in Figure 3. This information can be used to 'supervise' the behaviour of the fitness function and thus to tweak the appearance of the final virtual portrait (see Figure 5).

As mentioned in the introduction, our fitness function differs from those found in other GAs. In the classical approach, the function remains *fixed* during the whole computation time. In our algorithm, we use a *dynamic* fitness function which is affected by the evolution process. In every iteration, when a new set of portraits is evolved, it is used to update the set of images used by the eigenfaces algorithm (illustrated as the highlighted part of Figure 4), which causes the fitness function to change its values. We find such a function more appropriate for solving the problem of virtual portrait generation.

Before the results are presented, one peculiar aspect of the discussed problem should be noted which differentiates virtual portrait generation from most other optimisation problems. In our case, we are not simply interested in obtaining the fastest convergence to the optimal solution. In fact, such an optimal solution, *i.e.* a face image with the greatest degree of similarity to the historical figure's relatives, would be a simple 'average face' computed from the provided relatives' portraits (assuming no additional facial features are requested). This 'average face' could be generated with the morphing technique (described in Section 2.4), which would require neither an advanced optimisation algorithm nor significant computation resources.

Instead of taking this approach, we decided that our algorithm should incorporate some degree of randomness, which is also present in the natural processes which determine our facial appearance. As members of any family are never identical, but all have some characteristic, 'random' features, we do not focus on obtaining the 'ideal' result. Instead, we try to generate portraits which are similar to the provided relatives' images, but also have some random deviations. This is achieved through the use of the dynamic fitness function.

Figure 6 compares the fitness values obtained with a fixed and a dynamic fitness function (lower values mean greater similarity to the relatives' portraits). The values were normalised so that '1.0' denotes the value obtained for the initial population, *i.e.* the input set of historical portraits. From the perspective of classical optimisation problems, the fixed fitness function seems to perform better, as the obtained minimal value is significantly lower than in the dynamic case. In other words, a single 'optimal' portrait is created. In the dynamic case, the function's values change between iterations and so there is no strict convergence

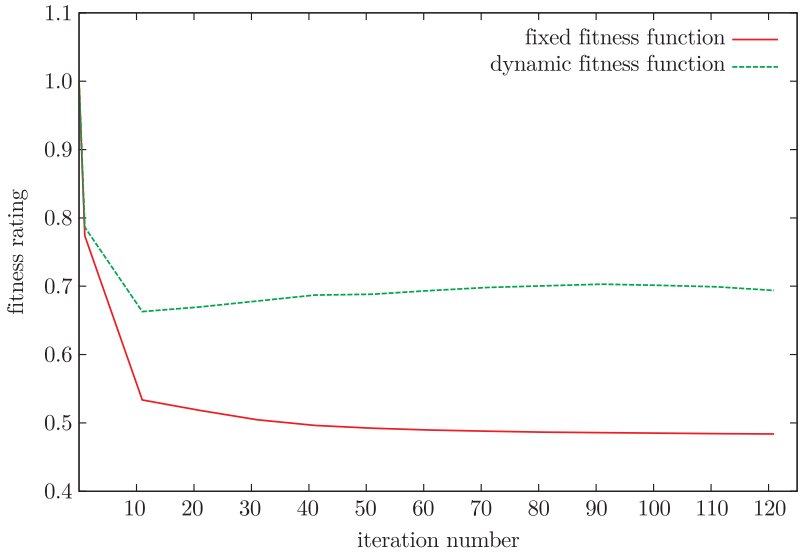

Figure 6. Fitness function values obtained for the fixed and dynamic case

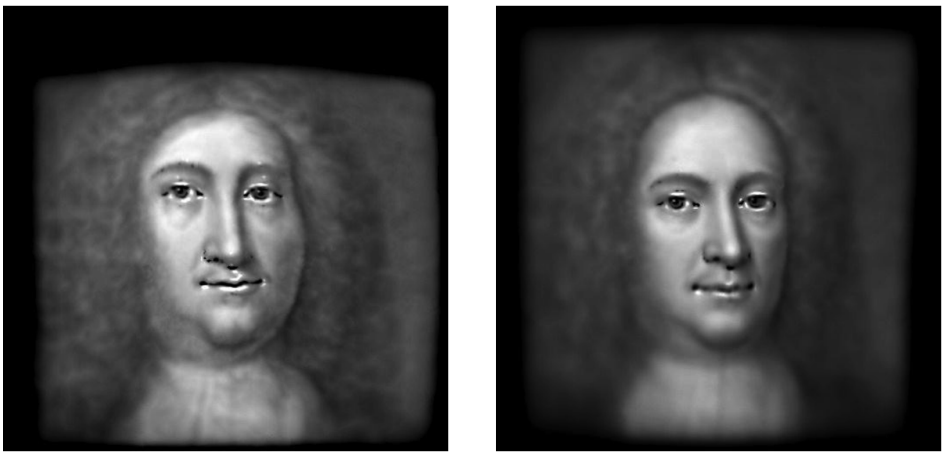

Figure 7. Results obtained using the fixed (left) and the dynamic (right) fitness function

to one optimal solution. Consequently, the results have higher (*i.e.* worse) fitness rating than in the case of the fixed function, but are also more varied.

Although such characteristics would probably discredit the dynamic fitness function in the classical optimisation problems, we find it appropriate to use it for virtual portrait generation. The algorithm using the dynamic function tends to produce more varied results which often also seem more realistic, *cf.* Figure 7. The introduced degree of randomness makes the algorithm more similar to the genetic and environmental processes occurring in nature. It should also be noted that our approach of using images generated in previous iterations allows us to overcome to problem of limited input data set, as it gives us a greater degree of freedom in manipulating the base portraits set used by the eigenfaces algorithm.

2.3. Selection

Face images are selected for recombination using a *probabilistic tournament selection*: a random set of faces is chosen for a *tournament*, and the faces with better rating are more likely to 'win' (*i.e.* be selected for recombination). The selection phase is similar to procedures found in other genetic algorithms and so will not be described here in detail.

The selection process can be controlled through standard parameters:

- *number of iterations* – the number of evolution iterations to conduct. If this parameter is too low, the result face images may not resemble the provided portraits of the historical figure's relatives. If the parameter is too high, the result images may be blurry or distorted;
- *survival rate* – determines how many of the *strongest specimens* (*i.e.* the highest rated images) should be kept for the next iteration. If too few specimens are kept, some highly rated images may be discarded;
- *breed size* – determines how many new images shall be generated every iteration. Increasing this value improves the degree of variety of images in the population (more images are created, each having some random mutations) but also increases the computation time;
- *tournament size* – number of images chosen for probabilistic tournament selection. Can be used to adjust *selection pressure*.

The specific values of the parameters were adjusted experimentally.

2.4. Recombination and mutation

Recombination and mutation are performed on the characteristic points identified on every processed face (*cf.* Figure 3). First, for each pair of faces selected for recombination, an 'average' set of points is computed. Then, slight random modifications are applied to the points, resulting in modifications of facial features (*e.g.* bigger eyes, narrower nose), see Figure 8.

As mentioned in the Introduction, the actual image of every newly evolved face is created with the *morphing* technique. The first step of the morphing procedure is to take each face from a recombination pair and *warp* the images so that their characteristic points match the ones generated in the previous step. After both images are transformed, they are blended together to produce a new portrait of a person similar to both of its 'parents', see Figure 9.

Additional adjustments can be introduced to the mutation process through various control parameters:

- *mutation strength* – determines the extent of introduced characteristic points random displacement and so affects the overall degree of mutation. Higher mutation strength increases face variety, but may also lead to significant image distortion. There are different mutation strength factors specified for different parts of the face. This means that the degree of allowable mutations differs between face regions. Choosing those factors incorrectly may lead to unrealistic face mutations;

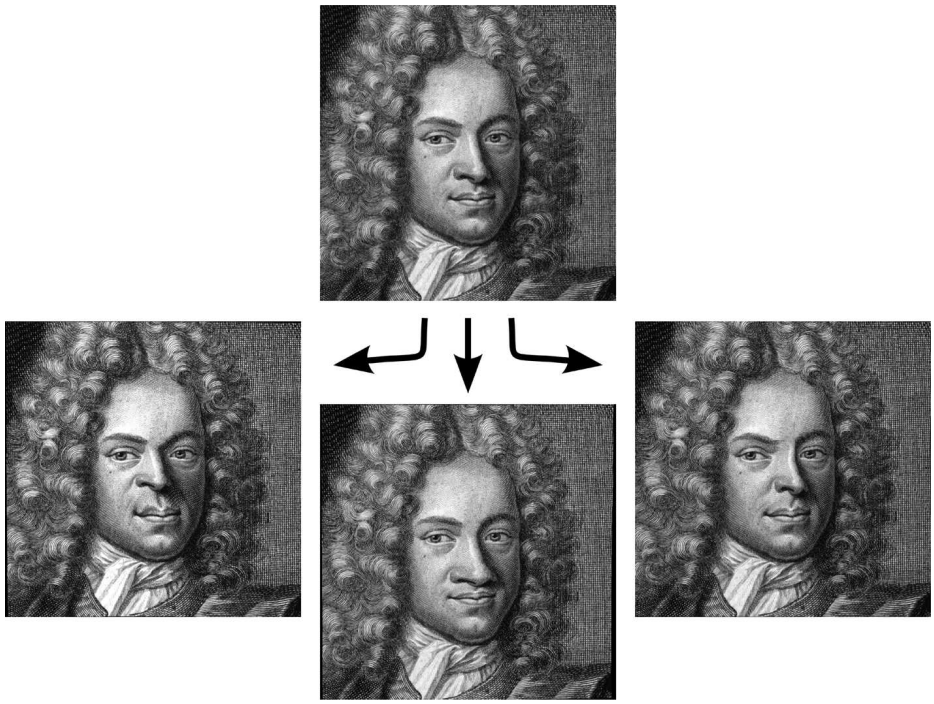

Figure 8. Examples of nose mutations (effect exaggerated for presentation purposes)

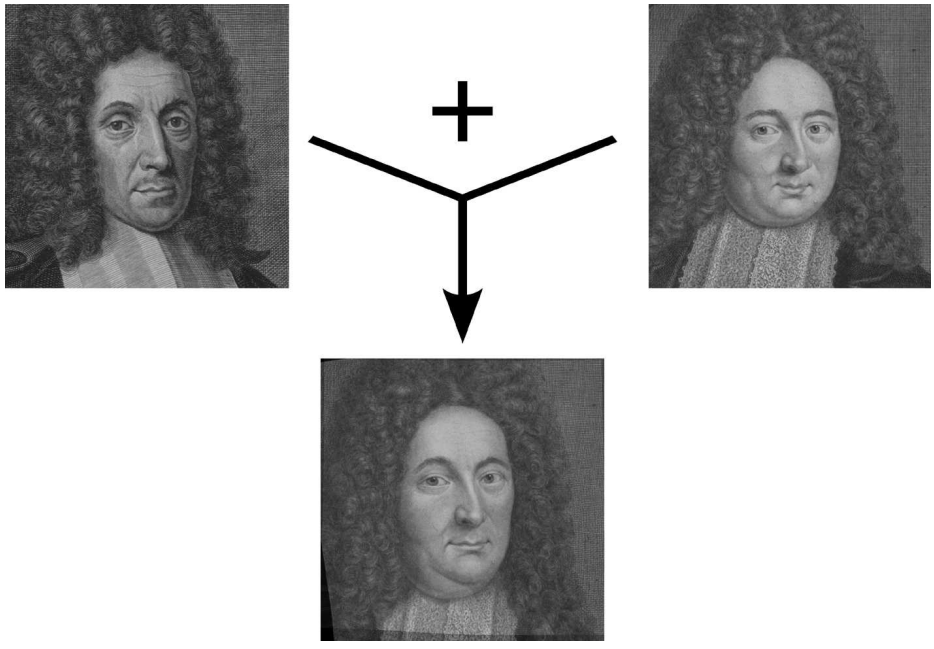

Figure 9. Face morphing – the intermediate image (in the middle) is created from the other two and so resembles both of them

- *mutation damping* – determines the degree by which the mutation strength is reduced in every iteration. This reduction allows more subtle modifications to take place as the algorithm progresses;
- *sharpening strength* – the morphing procedure causes the resulting images to appear blurry, especially after many iterations. To prevent this effect, a sharpening filter may be used. A sharpening factor that is too high results in visible artifacts.

Similarly to the selection parameters, the values of the above parameters were chosen empirically.

2.5. Output

After each iteration, the algorithm proposes a set of ten portraits which were rated as the best from the last population (see Figure 10). As it is hard for a computer to judge facial appearance, the final choice of the most appropriate face is left to the personal judgement of a human operator.

Figure 10. Sample output images generated by the algorithm

As with many other genetic algorithms, the stopping criteria cannot be easily defined. During our experiments, we observed that after a number of iterations the results become increasingly similar and the visual appearance of the generated portraits does not improve with further iterations. Figure 11 shows a numerical measure of difference between images generated in consecutive iterations. The values were normalised so that '1.0' denotes the difference between the portraits generated during the first two iterations. We found that to obtain a satisfactory solution, our algorithm needs about fifty iterations, which corresponds to relative differences of about 0.1. Further processing introduces only minor changes to the generated virtual portraits. If the number of iterations is set too high (over 150), the images appear blurry and distorted, which is a consequence of using the morphing algorithm.

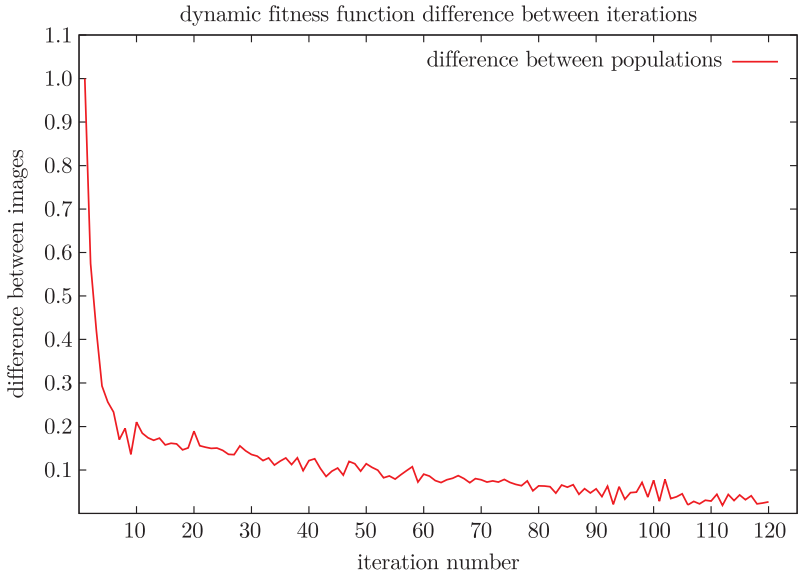

Figure 11. Differences in portraits generated in consecutive iterations

3. Example

As an example, we attempted to create a virtual portrait of a $17^{\text{th}}/18^{\text{th}}$ century scientist Daniel Gabriel Fahrenheit. Although his name is well-known because of his inventions (*e.g.* the popular temperature scale), there are no known paintings or drawings which would depict his facial appearance. Fortunately, a few portraits and statues of his relatives were available for us to use in our algorithm. We also used a set of portraits of people living in the same historical period. The process of creating Fahrenheit's virtual portrait is presented in Figure 12.

4. Distributed implementation for the KASKADA platform

Firstly, a sequential implementation of the algorithm was developed. As shown in Figure 2, there are four main processing stages:

- **A**: fitness function update,
- **B**: population rating,
- **C**: selection,
- **D**: recombination and mutation.

In our implementation, stages **B** and **D** can be parallelised, while stages **A** and **C** must be performed serially. The sequential algorithm was executed and the execution times of different program fragments were measured, as shown in Figure 13. The percentage P of execution time taken up by the parallelisable operations equals:

$$P = \frac{t_B + t_D}{t} \approx 99.74\% \tag{1}$$

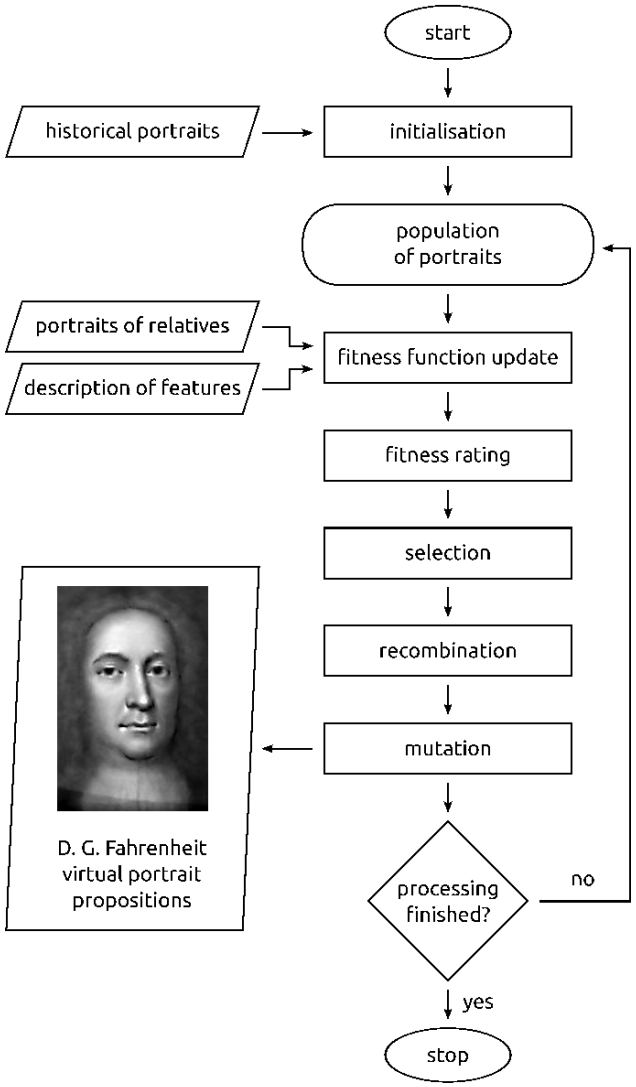

Figure 12. Creating a virtual portrait of D. G. Fahrenheit

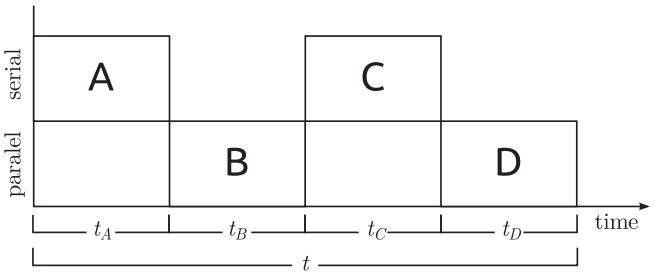

Figure 13. Execution time measurement for different stages of the algorithm

where t_A, t_B, t_C, t_D denote the time taken by the corresponding processing stages, and t is the total execution time, *cf.* Figure 13.

According to *Amdahl's law* (see [19, 20]), the speedup S_N gained from running an algorithm on N parallel processors is equal to:

$$S_N = \frac{1}{(1-P)+\frac{P}{N}} \tag{2}$$

In our case, the maximum potential speedup S_{MAX} (assuming $N \to \infty$) would be equal to:

$$S_{\mathrm{MAX}} = \frac{1}{1-P} \approx \frac{1}{1-0.9974} \approx 384.62 \tag{3}$$

Although the obtained S_{MAX} value is merely a theoretical one, it indicates that the computation time could be significantly reduced using parallel processing techniques.

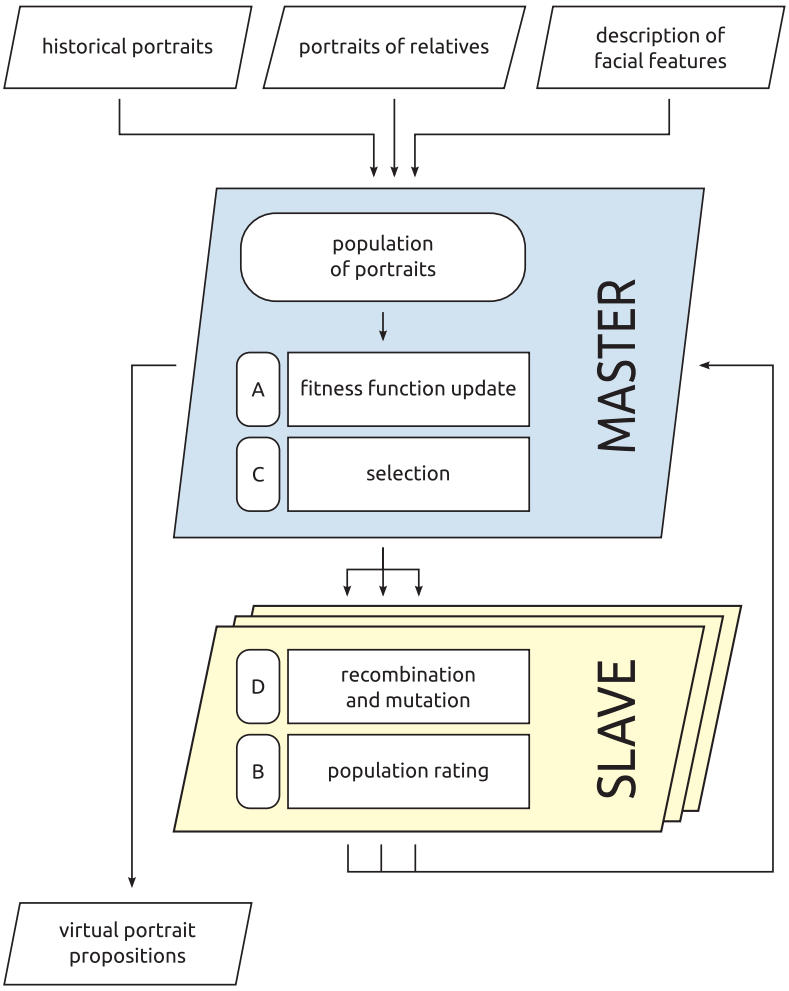

Figure 14. Overview of a *master-slave* implementation of the proposed algorithm

A parallel implementation of our algorithm was created for the KASKADA platform [16, 17]. The platform supports development and execution of distributed algorithms in two processing scenarios: *multimedia data stream processing* and *master-slave processing*. The second model can be successfully used for implementing genetic algorithms [14, 15]. Before the *master-slave* implementation of our algorithm could be developed, some of the processing stages had to be reorganised (see Figure 14). The *master* is responsible for all operations which must be performed globally, *i.e.* on the whole population. This includes algorithm initialisation, fitness function update (*i.e.* creation of new eigenfaces) and selection of face pairs for recombination and mutation. The *slave* part performs all tasks which can be parallelised, *i.e.* can be done in isolation from the whole population. These tasks are: face recombination and mutation and population rating. Most of the compute-intensive operations (especially face morphing) are performed by the *slaves* and hence a good degree of parallelisation is achieved.

Tests of the implementation have been performed on the *Galera+* compute cluster [18] controlled by the KASKADA platform. Each of the cluster's nodes is equipped with two Intel Xeon L5640 CPUs and 16 GB of RAM. Each of the processors contains 6 physical computing cores equipped with the *Intel Hyper-Threading* technology (see [21]) thanks to which every physical core is interpreted as two logical cores by the operating system. Thus, on every computing node there are 24 logical CPUs available.

Performance results obtained for the parallel implementation of our algorithm running on a single computing node are presented in Figure 15. By $t_{n:p}$ we denote the execution time of a single iteration of our algorithm on n compute nodes, every node running p parallel threads. A single iteration of a serial version of the algorithm took $t_{1:1} = 14038$ seconds. For up to 13 threads, the observed improvement in performance was nearly linear. The minimum iteration time for a single node was achieved for 22 threads and was equal to $t_{1:22} = 1087$ seconds.

Next, the performance was measured for multiple compute nodes. The results are presented in Figure 16. One node is always used for the *master* process and remains idle most of the time, while the others execute the *slave* processes which perform most of the computations. Because of this, only a very limited improvement is gained from running the algorithm on two nodes. For a larger number of nodes, a significant improvement can be observed. The shortest iteration time was obtained for 48 nodes and was equal to $t_{48:22} = 45$ seconds. By comparing this value with the time achieved by the sequential implementation, we can compute the actual speedup S_{actual}:

$$S_{\text{actual}} = \frac{t_{1:1}}{t_{48:22}} \approx \frac{14038}{45} \approx 311.96 \tag{4}$$

5. Final remarks

We presented a genetic algorithm for generating virtual portraits of historical figures whose facial appearance is unknown. Historical portraits from the

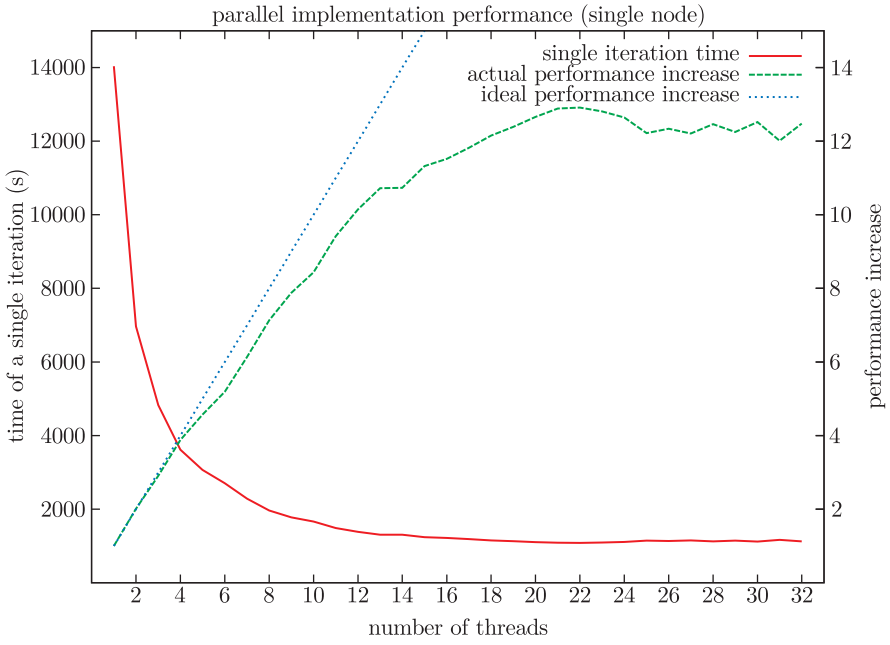

Figure 15. Iteration times and performance improvement for different numbers of parallel threads (single computing node)

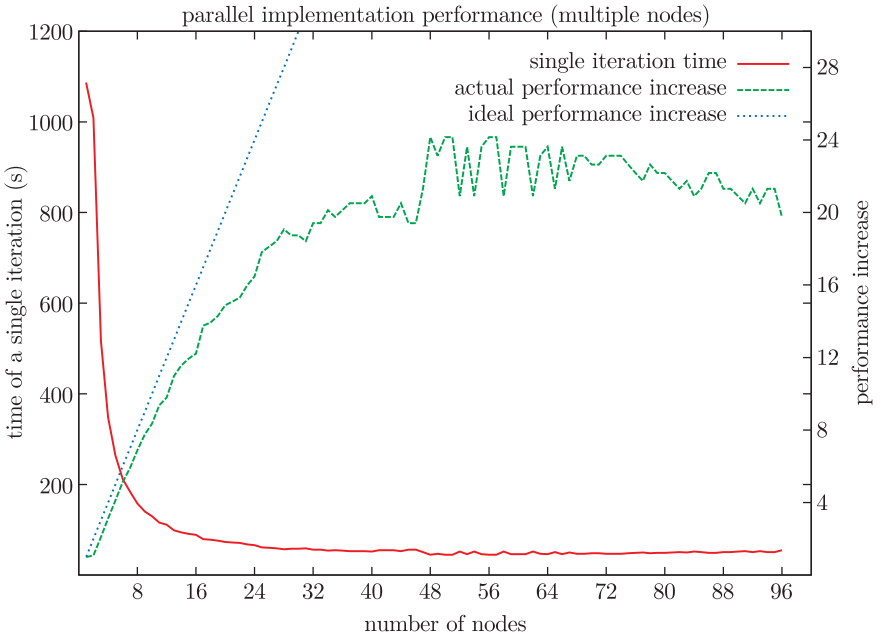

Figure 16. Iteration times and performance improvement for different numbers of computing nodes

appropriate period are used as a basis for the algorithm to ensure a proper general appearance of the generated images. The output images are rated by comparison with portraits of family members and description of specific facial features. A new dynamic type of fitness function was created to overcome the problem of limited number of available input images and improve the quality of the generated portraits. A parallel implementation was developed for the KASKADA platform and it was established that a high speedup can be obtained when executing the algorithm in a high-performance compute cluster environment.

It must be noted that we cannot claim our algorithm to produce scientifically accurate results. Genetic and environmental processes which influence human appearance are still largely a mystery to us and so are practically impossible to simulate. Our algorithm should be seen only as a first step in solving the problem of creating virtual portraits.

We recognise the possibility of future improvements to our algorithm which could be achieved by utilising medical and biological theories concerning the facial appearance of human beings. Furthermore, possible future advancements in these areas may make the process of re-creating realistic virtual portraits easier to model and simulate. It is also an interesting question if our dynamic fitness function can be adapted and used in solving other classes of problems.

Acknowledgements

This work was carried out as a part of the MAYDAY EURO 2012 project, Operational Program Innovative Economy 2007–2013, Priority 2 "Infrastructure area R&D".

References

[1] Whitely D 1993 *A Genetic Algorithm Tutorial, Technical Report CS-93-103*, Colorado State University
[2] Konak A, Coit D W and Smith A E 2006 *Reliability Engineering & System Safety, Genetic Algorithms and Reliability* **91** (9) 992
[3] Özyer T, Liu Y, Alhajj R and Barker K 2004 *Proc. Third Int. Conf. Advances in Information Systems, ADVIS'04*, Springer-Verlag Berlin, Heidelberg, pp. 451–461
[4] Bevilacqua V, Pacelli V and Saladino S 2011 *Proc. 7th Int. Conf. Advanced Intelligent Computing, ICIC'11*, Springer-Verlag Berlin, Heidelberg, pp. 186–193
[5] Mishra B S P, Addy A K, Roy R and Dehuri S 2011 *Proc. Int. Conf. on Communication, Computing & Security, ICCCS'11*, ACM New York, USA, pp. 409–414
[6] Toffolo A and Benini E 2003 *Evolutionary Computation* **11** (2) 151
[7] Linton R C 2004 *Proc. 42nd Annual Southeast Regional Conf., ACM-SE 42*, ACM New York, USA
[8] Turk M and Pentland A 1991 *J. Cognitive Neuroscience* **3** (1) 71
[9] Lee Y H, Han C W and Kim T S 2006 *Proc. 6th Int. Conf. on Computational Science, ICCS'06*, Springer-Verlag Berlin, Heidelberg, **IV**, pp. 862–869
[10] Wolberg G 1998 *Visual Computer* **14** 360
[11] Zhang Z, Wang L, Guo B and Shum H Y 2002 *Trans. on Graphics, Proc. ACM SIGGRAPH*, ACM New York, USA, **21** (3) 457
[12] Kekre H B, Sarode T K and Patil S M 2011 *Proc. Int. Conf. & Workshop on Emerging Trends in Technology, ICWET'11*, ACM New York, USA, pp. 357–362

[13] Bookstein F L 1989 *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11** (6) 567
[14] Cantu-Paz E 1998 *Genetic Programming*, Madison, USA, pp. 455
[15] Eklund S E 2004 *Parallel Computing* **30** (5-6) 647
[16] Krawczyk H and Proficz J 2012 *Interactive Multimedia*, InTech, Croatia, pp. 289–312
[17] Krawczyk H and Proficz J 2010 *Proc. Int. Conf. on Signal Processing and Multimedia Applications, SIGMAP*, SciTePress, Athens, Greece, pp. 26–31
[18] CI TASK 2012 *Galera Plus*, http://www.task.gda.pl/kdm/kdm/gplus
[19] Jordan H F and Alaghband G 2003 *Fundamentals of Parallel Processing*, Prentice Hall, Upper Saddle River, USA
[20] Sun X H and Chen Y 2010 *J. Parallel and Distributed Computing* **70** (2) 183
[21] Marr D T, Binns F Hill D L, Hinton G, Koufaty D A, Miller J A and Upton M 2002 *Intel Technology J.* **6** (1) 4