# Natural MPEG-4 Facial Expression using Genetic Algorithms

Lijia Zhu and Won-Sook Lee

***Abstract*— The MPEG-4 standard specifies feature points (FPs) and a set of facial animation parameters (FAPs) associated with the neutral face model. Given motions of MPEG-4 FPs, it is the developer's freedom to implement facial animation, meaning motion vectors for surface points of the 3D facial mesh. Our goal is to produce natural-looking motions for all surface points by specified MPEG-4 FP motions. We make use of an existing expressive animation database of limited size and we propose a method which uses Genetic Algorithms (GA) in interpolation and extension of the given expression data. The resulting animation has the same quality as the original animation data. Moreover, it covers a wider expression space after learning examples in the bank via GA.**

***Index Terms*— Facial animation, Genetic Algorithms, Interpolation, MPEG-4**

## I. Introduction

The MPEG-4 standard for facial animation came into being in 1999. Pandzic and Forchheimer [1] give a very detailed introduction on it. MPEG-4 specifies a neutral face model, a number of feature points (FPs) on this neutral face and a set of facial animation parameters (FAPs). FAPs specify the motions of the FPs. Given motion vectors of the FPs, it is the developer's freedom to implement motions for all surface points of the neutral face. Many approaches have been proposed to implement it. Noh *et al.* [2] use Radial Basis Functions (RBF) to create facial animation based on movements of control points. Garchery *et al.* [3] compute facial deformation using FPs only. The aforementioned techniques belong to the category of feature-based approaches, which calculate motions for all vertices of the face mesh based on FP movements. However, it is difficult to animate the face in a purely geometrical way to produce cheek deformation and sophisticated wrinkles.

L. Zhang *et al.* [4] use a capture system which records videos of a moving face. Then they propose an approach that goes from video sequences to high-resolution animated face models. The resulting 3D meshes illustrate very high-quality facial animation and in addition the meshes are consistently parameterized. With this expressive data available, it would be very beneficial to make use of the data and interpolation is probably the most suitable technique for this case. There are many researches on interpolation (e.g., [5] and [6]) for producing facial expression.

Our focus here is on producing various expressions for one subject with all surface points for high quality as well as using a set of FPs for easy control. The input of our system is 3D motion vectors for FPs of the neutral face model. In this poster we produce facial expressions by interpolating and extending expression models in the expression bank. We propose a novel approach to find the interpolation weights implicitly via Genetic Algorithms (GA). Essentially, GA's problem solving approach is to evaluate possible solutions at each generation, and then the offspring solution is generated to update the population. The fitter individuals are encouraged to survive and contribute more to the population. As a result, the optimal solution progressively improves during the process of evolution.

## II. Preparation

Figure 1 **:** Sample expression models in the bank

Our first step is to construct the expression bank containing several key expressions. In this poster, we make use of the existing animation data produced by L. Zhang *et al.* [4]. Their animation result can be found in [7]. There are a total of 384 animated models. They all share the same vertices and structures. We select 30 key models from their data to construct our bank (Figure 1).

For defining FPs in the neutral face model, we respect the MPEG-4 standard. MPEG-4 defines a total of 84 FPs. The FP set we defined (Figure 3a), which includes 56 FPs, is the subset of MPEG-4 FP set.

## III. Our GA Approach

In our GA approach, the population is initialized with the previously constructed expression bank. The population size is 30. The population size remains constant during our overall GA process. In each generation, we randomly evaluate three expressions in the bank. Now we need to design the fitness function for evaluation. Let us consider any one FP now. Suppose $V_a$ is the FP motion vector which is specified in the input requirement and $V_b$ is the vector for that FP in the selected expression face. It is better to have a smaller angle $\theta$ value between those two vectors. It hints that it is better to have

L. Zhu and W. -S. Lee are with School of Information Technology and Engineering, University of Ottawa, 800 King Edward Ave., Ottawa, Ontario, K1N 6N5, Canada (phone: 1-613-562-5800 ext. 2501; fax: 1-613-562-5664; e-mail: {lzhu058, wslee}@uottawa.ca).

a larger value of cos$\theta$. The cos$\theta$ value can be expressed with these two vectors as: $(V_a \bullet V_b) / (|V_a| * |V_b|)$. We can sum up cos$\theta$ values for each FP. Then we normalize the total sum with the total number of FPs. The best value that can be achieved is 1. However, 1 is just the ideal value. Our aim is to produce optimal value which is as close to 1 as possible. The closer the value is to 1, the greater the similarity between the target expression according to the input requirement and the selected expression.

We use our fitness function to evaluate the three previously selected expression faces. The individual with the worst fitness value in this tournament is not allowed to survive in the next generation. We remove it from the population and replace it with the offspring of two winners. The offspring is produced by averaging the two winners. Then we adjust FP motion vector lengths of the offspring to meet the requirement. We sum up motion vector lengths for FPs in the system input. Also we sum up vector lengths for those FPs in the offspring. Then we use the scale factor between those two values to adjust the motion vectors for all surface points of the offspring.

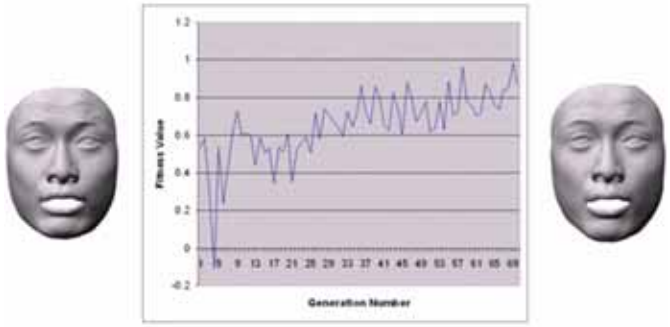


Figure 2: (a) Original model from [7] but is not the expression model in our expression bank; (b) Visualize the evolution of the fitness values; (c) The output of our GA system.

Finally, the offspring is used to replace the worst individual in the previous tournament. The original population now comes to the next generation. We iterate such process again and again until 70 generations are processed. How the population converges to the optimized solution can be visualized in Figure 2b where X-axis denotes the generation number and Y-axis shows the fitness value for the generated offspring. We keep track of the best individual over all generations in the whole GA process. As GA process terminates, the best individual expression face so far is returned as our GA system output (Figure 2c). We extract the motion vectors for FPs in Figure 2a and then get motions for all surface points by learning the examples in the bank. As you can see, our system output Figure 2c is quite similar to the original expression in Figure 2a.

Further improvement to our system is to decompose the face into more regions. Many researches (e.g., [5], and [8]) split the face into several regions for better control of animation. We divide the face into four sub-regions (Figure 3b). It also offers the possibility of producing asymmetric expression which makes the resulting facial expressions more natural. Given 3D motion vectors for FPs in the system input, we decompose it into four sub-goals. We apply GA to search solutions in the expression bank with four sub-goals separately. Then we combine four optimized sub-solutions together to produce the synthesized facial expression.

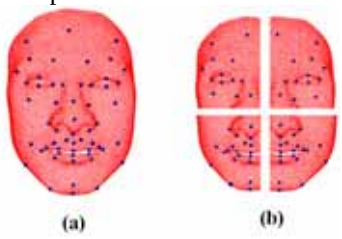


Figure 3: (a) FPs defined in our system; (b) The face is decomposed into four regions.

## IV. Results

Using our GA technique, we are capable of producing very natural-looking novel expressions. Figure 4 shows our facial expression results. We use GA to learn examples from our expression bank. A wider range of novel expressions can thus be produced from the existing database of limited size.

Figure 4: Facial expression results via GA.

## Acknowledgment

We would like to thank Li Zhang and Steven M. Seitz in the Graphics and Imaging Laboratory of the University of Washington for allowing us to use their face animation data.

## References

[1] I. Pandzic and R. Forchheimer. *MPEG-4 facial animation: the standard, implementation and applications*. Wiley, 2002.

[2] J.Y. Noh, D. Fidaeo and U. Neumann. Animated deformations with radial basis functions. *Proc. ACM symposium on virtual reality software and technology*, Seoul, Korea, 2000, pp. 166-174.

[3] S. Garchery, A. Egges and N. Magnenat-Thalmann. Fast facial animation design for emotional virtual humans. *Measuring Behaviour*, Wageningen, NL, Sept. 2005.

[4] L. Zhang, N. Snavely, B. Curless, and S.M. Seitz. Spacetime faces: high-resolution capture for modeling and animation. In *ACM SIGGRAPH Proceedings*, Los Angeles, CA, Aug. 2004.

[5] C. Kouadio , P. Poulin and P. Lachapelle. Real-time facial animation based upon a bank of 3D facial expressions. *Proceedings of the Computer Animation*. page128, June 1998.

[6] E. Chuang and C. Bregler. Performance driven facial animation using blendshape interpolation. *Standford University Computer Science Technical Report*. CSTR-2002-02, Apr. 2002.

[7] Spacetime Faces. http://grail.cs.washington.edu/software-data/stfaces/index.html.

[8] D. Fidaleo and U. Neumann. Analysis of co-articulation regions for performance driven facial animation. *Journal of Computer Animation and Virtual Worlds*, 2004,15: pp. 15-26.