Facial Expression via Genetic Algorithms

Lijia Zhu and Won-Sook Lee
School of Information Technology and Engineering, University of Ottawa
800 King Edward Ave., Ottawa, Ontario, K1N 6N5, Canada
{lzhu058, wslee}@uottawa.ca

Abstract

The MPEG-4 standard specifies feature points and a set of facial animation parameters (FAPs) associated with the neutral face model. Given motion vectors of MPEG-4 feature points (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 animation vectors 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 novel approach which uses Genetic Algorithms (GA) in interpolation and extension of the given expression data. Firstly, we construct the expression bank based on the available animation data. Then given sparse motion vectors of the FPs, we utilize GA to implicitly assign weights to examples in the bank and thus produce realistic optimized result after GA evolution. After introducing our initial GA approach, we present our improvements on producing natural-looking novel expressions by decomposing the face into several regions. 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.

Keywords: Facial animation, MPEG-4, FAPs, Genetic Algorithms, Interpolation

1. Introduction

Producing realistic facial animation is a challenging task for researchers in the computer animation field. Since the pioneering work of Parke ([1] and [2]), many different approaches have been proposed for producing facial expressions. Commonly, facial animation is more easily controllable with a set of feature points rather than all surface points. Here facial feature points are the prominent points on the face that capture the distinguished features of the face. The assumption is that the motions of feature points are capable of carrying enough information to drive all surface points for producing animation.

In 1999, MPEG (Moving Picture Experts Group) developed the ISO/IEC standard called MPEG-4. Pandzic and Forchheimer [3] give a very detailed introduction and discussion on MPEG-4 facial animation. MPEG-4 specifies a neutral face model, a number of feature points (FPs) on this neutral face as reference points and a set of facial animation parameters (FAPs), each corresponding to the specific facial action deforming a face model in its neutral status. 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

model. Many approaches have been proposed to implement it. Noh et al. [4] use Radial Basis Functions (RBF) to create facial animation based on movements of control points. Kshirsagar et al. [5] compute regions influenced by each of the control points and the corresponding weight for deformation for all the vertices in the influence region. Then from the displacement of the FPs, they calculate the actual displacements for all vertices of the facial mesh. More recently Garchery et al. [6] also compute facial deformation using MPEG-4 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.

On the other hand, facial animation can be driven by motion vectors of all surface points of the neutral model. From this point of view, it is a surface-based approach. Usually, it produces more sophisticated and expressive facial expressions than the purely geometrical approach for surface deformation. Noh and Neumann [7] present the expression cloning system that maps motions of all surface points from a source to a target model. L. Zhang et al [8] use a capture system which employs synchronized video cameras and structured light projectors to record 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 animation data available, it would be very beneficial to make use of the data and interpolation is probably the most suitable animation technique for this case. Given a set of key facial expressions from existing animation database, a blend shape model can be constructed by considering every facial expression as a linear combination of key expressions. By varying the weights of the linear combination, a wide range of facial expressions can thus be produced.

There are many researches on interpolation for producing facial expression. Kouadio et al. [9] minimize the Euclidean distance between corresponding points and markers to obtain a linear combination of the basic expressions. Chuang and Bregler [10] produce facial animation with a combination of motion capture data. Weights for interpolation are calculated based on the least square solution. Chai et al. [11] find the K closest examples in the motion capture database and then linearly interpolate the corresponding examples. Pyun et al. [12] evaluate the weights for key models with cardinal basis functions, which consist of linear and radial basis functions. This allows them to obtain the output model by blending key models with those weights. Joshi et al. [13] propose an automatic, physically-motivated segmentation that learns the controls and parameters from blend shapes.

Our previous paper [14] shows how to perform expression cloning for a detailed surface from one person to another with laser-scanned faces. Our focus here is on producing various expressions for one subject using a surface-based approach for high quality as well as using a set of feature points for easy control. The input of our system is 3D motion vectors for MPEG-4 FPs of the neutral face model. In this paper 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). GA is inspired by the process of natural evolution [15]. 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 pool. Given the evaluation function, 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.

This paper consists of six sections. In section 2, we construct the expression bank and define the feature points for the neutral face model. Our GA approach is proposed in section 3 and then we show experiment results in section 4. Section 5 explains our

enhancements for the initial GA system. Finally we conclude our paper in section 6.

2. Preparation

Ekman and Keltner [16] propose six basic emotional facial expressions (joy, anger, fear, disgust, sadness and surprise). However, in the real world, the human face is capable of generating far more expressions than the basic ones. Many recent researches (for example, [17] and [18]) explore the methods to generate facial expressions for a continuum of pure and mixed emotions of varying intensity.

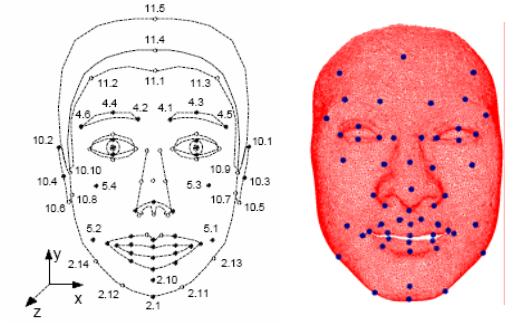
In this paper, we aim to produce diverse expressions via GA with the transition of the predefined emotional facial expressions in the expression bank. The desired emotional expressions are specified by motion vectors for the FPs. Then by our GA approach, we deform the surface with the motions for all the surface points by learning the examples in the bank.

Figure 1 **:** Dominant expression models

Our first step is to construct the expression bank containing several key expressions. In this paper, we make use of the existing animation data produced by L. Zhang et al [8]. Their attractive animation result can be found in [19]. There are a total of 384 animated models. They all share the same vertices and structures (23K vertices and 46K triangles) and there are six dominant emotional expressions (Figure 1) and others are intermediate expressions. We select 30 key models (including 6 dominant expressions) from their animation data to construct our expression bank. Given a set of expression models in the expression bank, the interpolated expression model E can be expressed as:

$$E = \sum_{i=1}^{n} w_i E_i$$

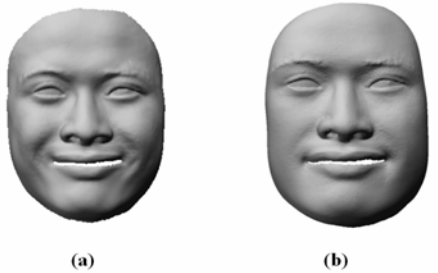
where w_i is the weight assigned to the specific expression face E_i in the expression bank, and n is the number of expression models in the bank. Based on the expression bank, a wide range of facial animation can be produced by varying w_i.



(a) **(b)**

Figure 2: (a) MPEG-4 feature point set; (b) Feature points defined in our system.

For defining FP positions in the neutral face model, we respect the MPEG-4 standard. MPEG-4 defines a total of 84 FPs (Figure 2a). The feature point set we defined (Figure 2b), which includes 56 FPs, is the subset of MPEG-4 feature point set. We exclude some MPEG-4 FPs which are in the areas of the tongue, tooth, ear and hair because our expression models in the expression bank do not include those parts. We also exclude some FPs where they are too close to each other. (E.g. some points in the eye area).



(a) (b)

Figure 3: (a) Expression model from [19]; (b) RBF global deformation driven by FP motions.

Given the motion vectors of MPEG-4 FPs, our goal is to produce natural-looking animation vectors for all the surface points. Originally, we plan to achieve this goal by RBF which is similar to the approach

proposed by Noh et al. [4]. We use motion vectors of FPs in the animated model (Figure 3a) to globally deform the neutral face model by our RBF library. Our previous work [14] successfully uses our RBF library to globally adapt the generic model to a laser-scanned face. However, for this case, our experiment result is not good (Figure 3b). First, the global shape is not correct because of a lack of FPs in the face outlines. In [14], we defined more feature points. The FP definition is very critical for result. Second, even if we adjust FP definition now and apply RBF locally, the global shape is correct, the skin deformation on the cheeks, however, will not be natural as it creates less folding than natural expression. So we decide to investigate a novel method to produce expressions based on surface points.

3. Our GA approach

Because GA is good at solving complex optimization problems, its use has become more and more popular in the computer graphics and computer animation fields. For example, Tohka [20] presents the approach that applies GA to optimization of deformable surface meshes; Bui et al. [21] use GA to adjust feature points on a target face to minimize the difference between the surface of the morphed face and the target face.

```
t = 0;
initialize a population P(t) of expression faces;
while (not termination condition) do
   randomly select 3 expression faces from P(t);
   evaluate three picked expressions;
   generate offspring and update P(t);
   t = t+1;
end while
```

Figure 4: Our GA structure

In this section, we aim to find the optimized weights in order to interpolate expression models in the expression bank. The basic structure of our GA approach is listed in Figure 4. The population of expression faces is initialized with the previously constructed expression bank. The population size for our GA is 30. The population size remains constant during our overall GA process.

In each generation, we randomly select 3 expression faces from the population. If we evaluate 2 expressions at a time, the GA process will converge more slowly. By using number 3, we will get rid of bad genes in the expression bank faster. On the other hand, we could evaluate more than 3 faces and update more faces in each generation. GA will converge faster. However it may lead to the local maximum solution. Therefore we decide to evaluate 3 expression faces at a time. Now we need to design the fitness function to evaluate randomly selected three expressions in the bank. The purpose of the evaluation is to determine how well the specific expression matches the system input (i.e. 3D motions for FPs).

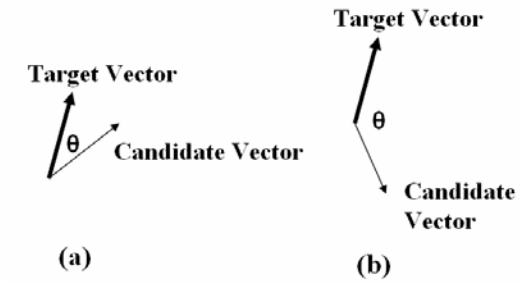


Figure 5: Illustration for explaining fitness function.

Suppose that there are a total of M feature points with non-zero motion vectors in the system input where $0 \leq M \leq 56$ because totally we define 56 feature points in our system. First, let us have a look at any FP with non-zero motion vector (Figure 5). Suppose that the target vector is the FP motion vector which is specified in the input requirement and the candidate vector is the vector for that FP in the selected expression face. Obviously, the candidate vector in Figure 5(a) is more qualified than that in Figure 5(b) according to the input requirement. It is better to have a smaller angle θ value between those two vectors. The angle θ can be expressed in terms of the target vector V_a and the candidate vector V_b :

$$\cos\theta = \frac{V_a \bullet V_b}{|V_a| * |V_b|}$$

It hints that it is better to have a larger value of cosθ for each feature point. We can sum up all cosθ values for all FPs with a non-zero motion vector in the input requirement. Then we normalize the total sum with the M value (Because the maximum total sum is: $M*cos0^0 = M$). Our fitness value F can thus be given by

$$F = \frac{\sum_{i=1}^{M} \cos\theta i}{M}$$

where M is the number of FPs with a non-zero target motion vector. The best value F 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 output 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. There are various crossover strategies to produce offspring from parents. The crossover operator plays a central role in GA [22]. The simplest crossover we used is called Midpoint crossover where the offspring lies exactly on the middle of two parents. Suppose P_1 and P_2 are the parents, the offspring is expressed as:

$$\text{Offspring} = 0.5*P_1 + 0.5*P_2$$

In Line crossover (a.k.a. Flat crossover), the offspring, which is located in a random position between the parents (P1 and P2), is given by:

$$\text{Offspring} = t * P1 + (1-t) *P2$$

where $0 \le t \le 1$. BLX-α crossover is the extension of Line crossover. The offspring can be calculated based on the parents (P1 and P2) in the following way:

$$\Delta = \alpha\,(P2-P1);$$
$$\text{Offspring} = t*(P1 - \Delta) + (1-t)*(P2 + \Delta)$$

where $0 \le \alpha \le 1$ and $0 \le t \le 1$. Line crossover is the special case of BLX- α crossover when t=0.5, and Midpoint crossover is also the special case of Line crossover when α=0 and t=0.5. Each of these three crossovers is capable of generating offspring from previous two winners in the tournament and thus leading to the optimal solution finally. We will discuss more on experiments with these 3 crossovers in section 4.

However, we need one more step to adjust the offspring. Previously, in the fitness function, we only consider vector directions of FPs. Now we need to adjust FP 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 the corresponding 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.

We need to mention that if we evaluate vector directions and lengths simultaneously in our evaluation function, we may fall in the dilemma to decide which of the following two cases is better: a) better directions but worse lengths; b) worse directions but better lengths. By treating vector direction and length separately as we proposed, we ensure that we won't be in such a dilemma.

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 N generations are processed. It is our GA termination condition. According to our experience, N=70 is chosen for the current system. It is decided by the population size and our strategy to refine the population. How the population converges to the optimized solution can be visualized in Figure 6b 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. When GA process meets our termination condition,

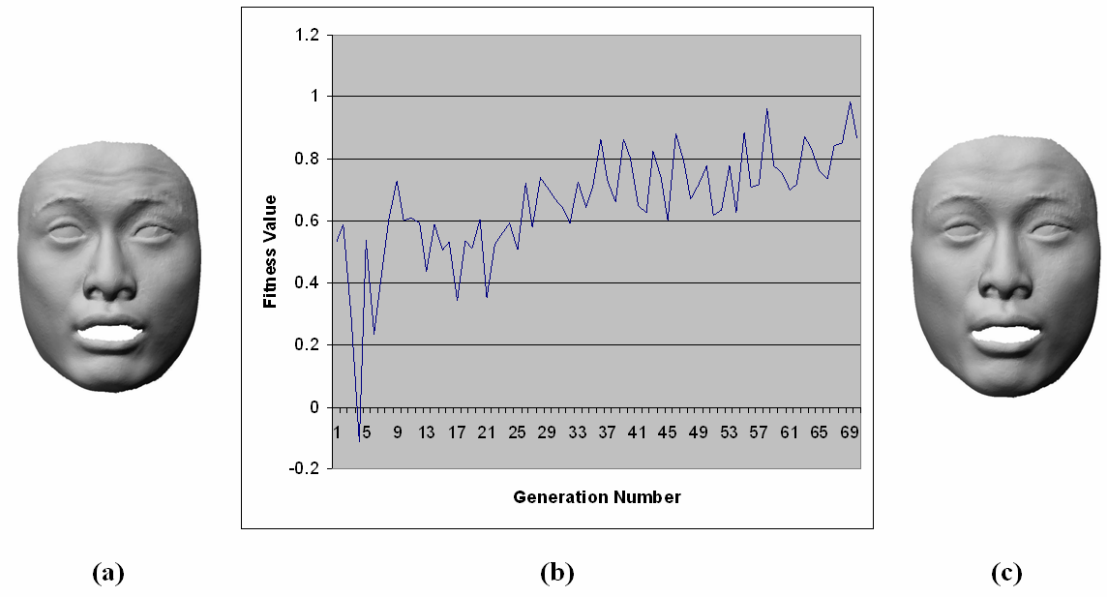


Figure 6: (a) Original model from [19] but is not the expression model in our expression bank; (b) Sample experiment result by using Line crossover; (c) The output of our GA system.

the best individual expression face so far is returned as our GA system output (Figure 6c). We extract the motion vectors for FPs in Figure 6a and use it to get motions for all surface points by learning the examples in the bank via GA. Our system output Figure 6c is quite similar to the expression in Figure 6a.

Essentially, in our GA approach, the interpolation weights are assigned to each expression model in the expression bank implicitly over the generations. The expressions which are less relative to our input requirement are assigned less weights. The genes from the good expressions are assigned more weights and are accumulated to contribute to our final output.

4. GA experiments

There are a total of 384 animated models in the original data [19]. Each of the models is made of about 23K vertices and 46K triangles. We have tested our system with 20 randomly picked models which are not in our expression bank but from their data [19]. We only use 3D motion vectors for FPs for those models to drive our system. Our test goal is to see whether we can produce the similar animation vectors for all the surface points as the original data. The experiments are done on a 2.60 GHz AMD Opteron PC with 2.0 GB of RAM. The GA is terminated when 70 generations are evaluated. The computation time is about 40 seconds.

	Avg. of Avg. Best Fitness Value	Avg. of STD
BLX-0.2	0.962	0.0068
Midpoint	0.960	0.0057
Line	0.970	0.0075

Table 1: GA system experiments with various crossovers

In our experiments, we have tried various crossovers with those randomly picked models. For each crossover, we run our GA system 10 times for each model. The experiment results are shown in Table 1. Avg. of Avg. Best Fitness Value stands for the average of the average best fitness value. Avg. of STD is the average standard deviation for that crossover. For the BLX-α crossover operator, we tested parameter α

with various values: 0.1, 0.2, 0.3, 0.4 and 0.5. According to our experiment experience, we choose α = 0.2 for its better performance over others. The performances of three crossovers (Midpoint, Line and BLX- α) do not vary too much. No crossover dominates the others. Usually, the ideal operator depends on the model itself. We assume experiment results are distributed normally. By the property of the standard deviation, two standard deviations away from the average account for roughly 95% of the data. We can get the approximate performance range for each crossover from Table 1. Compared with Midpoint crossover (0.949 ~ 0.971) and BLX-0.2 crossover (0.948~ 0.976), Line crossover (0.955 ~ 0.985) performs a little bit better.

We would like to mention that our GA system terminates when 70 generations are generated. If we permit more generations in the evolution process, BLX- α may perform better because BLX- α is trying to explore the solution space in a wider way. However, the computation time will increase if we process more generations. The choice of the crossover operator is a compromise between accuracy and convergence speed.

5. Enhancements for facial expression

After testing our initial GA system, we have also made several improvements on it. Firstly in order to produce facial expressions of various intensities, we use emphasize level parameter *c* to control our system input. The input FP motion vectors are multiplied by the value *c* before starting the GA process.
Figure **7** shows different final GA system outputs by varying parameter *c*.

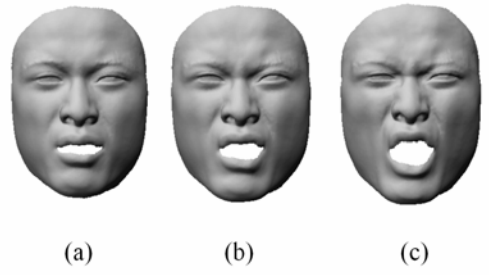


Figure 7: Emphasize level control parameter *c*. (a) *c* = 0.7; (b) *c* = 1.0; (c) *c*=1.4.

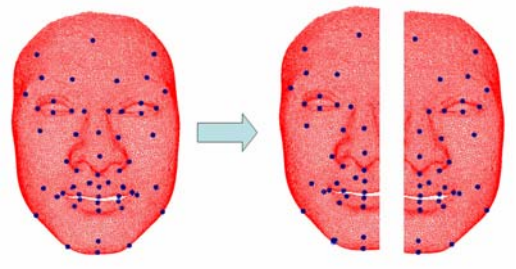


Figure 8: The face is decomposed into two regions.

One more improvement we made is to split the face into two parts: left and right faces (Figure 8). Two sub-regions overlap each other. It offers the possibility of producing asymmetric expression which makes the resulting facial expressions more natural. We apply GA for left and right FPs separately. Then we combine left and right face results together to produce a synthesized facial expression. However, special treatment is needed in the joint part where artefacts may appear. One possible solution is to apply a 3D smoothing filter for each point on the joint part (i.e. average motion vectors of the surrounding points). It works fine when the left and right face expressions do not differ too much. When the difference between the left and right faces is big, we will have unnatural expression result. Kouadio et al. [9] mention that points surrounding the borders should be included in both regions, and their new positions in each half expression are weighted accordingly.

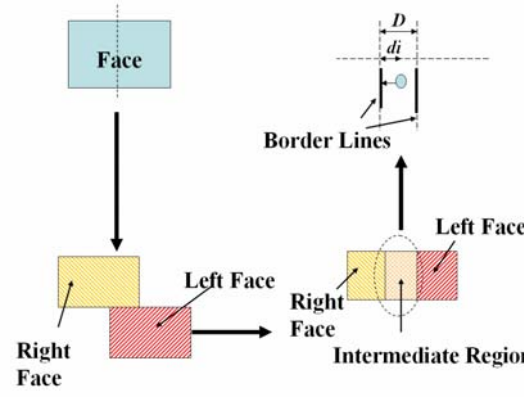


Figure 9: The strategy to remove the artefacts around the joint part for two sub-region decomposition.

Inspired by this idea, we propose the following sophisticated technique to tackle this issue (Figure 9). The left and right faces overlap each other around the central line. We design the intermediate region which blends the left and right face animation vectors with our distance-related weight assigning function. The formula for calculating the animation vector $V_i^{Intermedia\ te}$ for the *i*th point in the intermediate region is given as:

$$V_i^{Intermediate} = (1 - \frac{di}{D}) * V_i^{Right} + \frac{di}{D} * V_i^{Left}$$

where *di* is the distance of that point to the border line; *D* is the width of the intermediate region; V_i^{Left} is the motion vector for that point in the left face region and V_i^{Right} is that in the right face region. The left and right faces overlap each other in the intermediate region. They both contribute to the intermediate region. With our distance-related weight assigning function, there are no artefacts around the border lines. Figure 10 shows our asymmetric expression results by dividing the face into left and right regions.

Figure 10: Asymmetric expression results by dividing the face into left and right regions.

Further improvement to our system is to decompose the face into more regions. Many researches ([9], [10], [13], [23], [24] and [25]) split the face into several regions for better control of animation. Commonly, they divide the face region from two to ten sub-regions. We divide the face into four sub-regions (Figure 11). Firstly, we partition the face into left and right faces. Then for each half face, we divide it further into upper and lower parts. Each quarter region is controlled by the corresponding FP sub-group. 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. Similarly, the previously described technique for removing the artefacts in the joint parts is also applied.

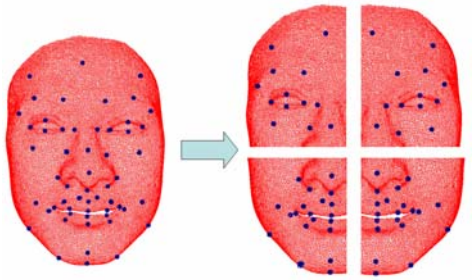


Figure 11: The face is decomposed into four regions.

For the ground truth models we tested in the previous section, we conduct experiments with Line crossover for 10 randomly selected models. Using our initial GA system, we have the average best fitness value 0.960. By decomposing the face into two regions, our average best fitness value is 0.972. The computation time increases from 40 seconds to two minutes on our previously mentioned PC. With the four sub-region scheme, we have the value 0.981. The computation time is about five minutes. It shows that, by dividing the face into several regions, we can improve results by finding several optimal sub-solutions via GA. However the computation time increases, it is the trade-off between accuracy and speed.

Figure 12: Facial expression results via GA (front view).

Figure 13: Different views of facial expressions results.

Using the above techniques, we are capable of producing more natural-looking novel expressions. Figure 12 and Figure 13 show our facial expression results produced after making the aforementioned improvements to our initial GA system. We use GA to learn examples from our expression bank given the sparse FP motions. A wider range of novel expressions can thus be produced from the existing database of the limited size.

6. Conclusion

In this paper we have presented the novel approach which uses GA in interpolation and extension of existing expressive animation data. Given sparse 3D motion vectors for FPs, we utilize GA to implicitly assign weights to examples in the expression bank and thus produce a realistic optimized result after GA evolution. After introducing our initial GA system, we present our improvements on producing very natural-looking expressions by decomposing a face into several regions. 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.

In our future research, we would like to use a motion capture device to capture facial motions with sparse markers. After that, we plan to use the captured data to produce animation by learning the existing animation data via GA. Then we can compare the ground truth (i.e. video-taped facial movements during motion capture session) with the reconstructed expressions.

Acknowledgements

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. We are also grateful to Andrew Soon for proof reading this document.

References

[1] F. I. Parke. Computer generated animation of faces. *Proceedings of the ACM annual conference*. Boston, Massachusetts, United States, 1972, pages 451 – 457.

[2] F. I. Parke. *A parametric model for human faces*. PhD Thesis, University of Utah, Department of Computer Science, 1974.

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

[4] 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, pages 166-174.

[5] S. Kshirsagar, S. Garchery, and N. Magnenat-Thalmann. Feature point based mesh deformation applied to MPEG-4

facial animation. In *Deformable Avatars. Norwell, MA: Kluwer*, 2001, pages 24–30.

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

[7] J. Y. Noh and U. Neumann. Expression cloning. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. Aug. 2001, pages 277-288.

[8] 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.

[9] 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 08-10, 1998.

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

[11] J. Chai , J. Xiao and J. Hodgins. Vision-based control of 3D facial animation. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation.* July 26-27, 2003, San Diego, California.

[12] H. Pyun , Y. Kim , W. Chae , H. W. Kang and S. Y. Shin. An example-based approach for facial expression cloning. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation,* San Diego, California, July 2003, pages 167 - 176.

[13] P. Joshi , W. C. Tien , M. Desbrun and F. Pighin. Learning controls for blend shape based realistic facial animation. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. July 26-27, 2003, San Diego, California.

[14] L. Zhu and W.-S. Lee. Modeling and Animating for the Dense Laser-scanned Face in the Low Resolution Level. *The 17th IASTED International Conference on MODELLING AND SIMULATION 2006*. May 24-26,2006. Montreal, Quebec, Canada

[15] David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley. MA, 1989.

[16] P. Ekman and D. Keltner. Universal facial expressions of emotion: An old controversy and new findings. In U. Segerstrale & P. Molnar (Eds.) *Nonverbal communication: Where nature meets culture.* pages. 27-46. Mahwah, NJ: Lawrence Erlbaum Associates. 1997.

[17] N. Tsapatsoulis, A. Raouzaiou, S. Kollias, R. Cowie and E. Douglas-Cowie, Emotion Recognition and Synthesis based on MPEG-4 FAPs. In *MPEG-4 Facial Animation: the standard, implementation and applications*. Pages 141-167. I. Pandzic and R. Forchheimer (eds), John Wiley & Sons, UK, 2002.

[18] I. Albrecht, M. Schröder, J. Haber and H. Seidel. Mixed feelings: expression of non-basic emotions in a muscle-based talking head. *Virtual Reality (Special Issue "Language, Speech and Gesture for VR")* pages 201-212. August 2005.

[19] Spacetime Faces. http://grail.cs.washington.edu/software-data/stfaces/index.html. Date of access: Apr. 2006.

[20] J. Tohka. Global optimization of deformable surface meshes based on genetic algorithms. *In Proc. of 11th International Conference on Image Analysis and Processing*, *ICIAP01*, pages 459 - 464, 2001.

[21] T.D. Bui, M. Poel, D. Heylen and A. Nijholt. Automatic face morphing for transferring facial animation, *Proc. 6th IASTED International Conference on Computers, Graphics and Imaging*. Honolulu, Hawaii, USA, August 2003, pages 19-23.

[22] F. Herrera, M. Lozano and J. Verdegay. Tackling real-coded genetic algorithms: operators and tools for the behaviour analysis. *Artificial Intelligence Review 12* (1998) pages 265 - 319.

[23] T.D. Bui, D. Heylen, M. Poel and A. Nijholt. Exporting vector muscles for facial animation. *Smart Graphics 2003*: pages 251-260.

[24] Q. Zhang , Z. Liu, B. Guo and H. Shum. Geometry-driven photorealistic facial expression synthesis. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. July 26-27, 2003, San Diego, California.

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