Multimed Tools Appl
DOI 10.1007/s11042-007-0184-x

An interactive facial expression generation system

Chuan-Kai Yang · Wei-Ting Chiang

Abstract How to generate vivid facial expressions by computers has been an interesting and challenging problem for a long time. Some research adopts an anatomical approach by studying the relationships between the expressions and the underlying bones and muscles. On the other hand, MPEG4's SNHC (synthetic/natural hybrid coding) provides mechanisms which allow detailed descriptions of facial expressions and animations. Unlike most existing approaches that ask a user to provide 3D head models, a set of reference images, detailed information of facial feature markers, numerous associated parameters, and/or even non-trivial user assistance, our proposed approach is simple, intuitive and interactive, and most importantly, it is still capable of generating vivid 2D facial expressions. With our system, a user is only required to give a single photo and spend a couple of seconds to roughly mark the positions of eyes, eyebrows and mouth in the photo, and then our system could trace more accurately the contours of these facial features through the technique of *active contour*. Different expressions can then be generated and *morphed* via the *mesh warping* algorithm. Another innovation of this paper is to propose a simple *music emotion analysis algorithm*, which is coupled with our system to further demonstrate the effectiveness of our facial expression generation. Through such an integration, our system could identify the emotions of a music piece, and *display* the corresponding emotions via aforementioned synthesized facial expressions. Experimental results show that in general the end-to-end facial generation time, from the time an input photo is given, to the time the final facial expressions are generated, is about 1 min.

C.-K. Yang (✉) · W.-T. Chiang
Department of Information Management,
National Taiwan University of Science and Technology,
No. 43, Sec. 4, Keelung Road, Taipei 106, Taiwan, ROC
e-mail: ckyang@cs.ntust.edu.tw

Keywords Facial expression generation · Active contour · Image morphing · Music analysis

Categories and Subject Descriptors C.3 [Special-Purpose and Application-Based Systems]: Signal Processing Systems

1 Introduction

As facial expressions are an important means for human beings to communicate with others, how to generate facial expressions by computers have interested many researchers. Due to the involved high complexity of the composition of expressions, facial expression generation is still considered a challenging task nowadays. Thanks to the invention of *motion capture* devices, human motion can now be realistically simulated on a 3D computer model. Inspired by such a trend, Gleicher et al. [4] proposed to *re-target* facial expression information captured from real persons to 3D virtual human models. However, to have a vivid look, the underlying anatomical knowledge on how bones/muscles affect facial expressions seems to be indispensable, thus increasing the difficulty of adopting such an approach. On the other hand, some related standards have also been developed. MPEG4's SNHC [7, 15] provides mechanisms that allow a basic model of face or body to be described and animated. Through such a compact description of related parameters, the desired high compression ratios could be achieved.

Given the fact that the generation of facial expressions is still non-trivial, we therefore set our goal to be the following. Given a photo of a person, our system could allow users to easily and interactively guide the feature extraction process with minimal manual effort, and the facial expressions of different emotions of the person could then be automatically generated. To prove the effectiveness and efficiency of our facial expression generation system, we further equip our system with a MIDI (musical instrument digital interface) music analysis ability, so that it could approximately determine the emotions presented by a music piece, and then "respond" by displaying the corresponding facial expressions accordingly. To have a smooth demonstration, the displayed expressions are *morphed* into animations by using the *mesh warping* technique. According to our experiments, the overall *end-to-end time*, i.e., from the time a photo is given for feature extraction, till the time the target facial expressions of different emotions are generated, is about 1 min.

To sum up, the main contribution of this paper is twofold. First, compared with most existing approaches, where 3D head models, a group of 2D reference images, a set of specific facial feature markers, numerous related parameters, and/or even non-negligible user assistance, are usually required from the users, our approach is relatively simple and intuitive: only a single photo and a few strokes are needed. Here the strokes are mainly used to mark three contours in the photo to approximately enclose all necessary facial features for ensuing facial expression generation. In comparison with previous approaches where user assistance is involved, ours is relatively easier, thus leading to better interactivity and efficiency. Second, as far as we know, we are the first to propose a framework on how facial expressions could be used to interact with music. While using MIDI music as a particular example, we

further develop a simple algorithm capable of analyzing music emotions, thus making the interactions between facial expressions and music possible.

The rest of the paper is organized as the following. Section 2 reviews the literature related to this work. Section 3 details how facial expression generation is done. Section 4 briefs the basic knowledge of MIDI and describes how the emotions of a MIDI music piece could be determined. Section 5 presents our synthesized results, and compares them with real expressions when applicable. Section 6 concludes this work and hints for potential future directions.

2 Related work

Facial expression generation has attracted many researchers since the early 1970s [11], and to have a more compact presentation, in this study we only review those papers that are more related to this work. Readers who are interested in learning more knowledge regarding this topic are referred to the excellent book written by Parke et al. [12]. As far as the output is concerned, it could be a 2D image [10, 15, 22, 29] or a 3D surface model [11, 13, 28]. And while some of the papers resort to physics-based or anatomy-based approaches [21, 24, 28], others choose to adopt geometry-based or image-based strategies [15, 22, 27, 29].

Parke [11] made the first attempt on facial expression synthesis by approximating a human face with roughly 250 polygons and 400 vertices, and how these polygons should be adjusted to convey different expressions was determined photogrammetrically with pairs of photographs. The transition between frames corresponding to different expressions is obtained with a cosine interpolation scheme to achieve an animation effect. Ekman et al. [2] proposed the first method to link facial muscles with facial expressions. By carefully examining videotapes of facial expressions, they defined the *Facial Action Coding System*, or *FACS* for short, to specify how each expression links with corresponding muscular contractions. For efficiency reasons, *action units* are used instead of muscles, as some action unit could be formed from more than one muscle, while sometimes the change caused by one muscle could lead to changes of more than one action unit. As differences exist among different individuals, the relationships between action units and expressions may also vary significantly, and as a result, manual assistance is usually required. Although the number of involved action units is just around fifty, the possible combinations could be more than a thousand. Based on FACS, Waters [24] identified the common parameters among different faces and applied them on a 3D surface model that allows a more general and flexible control of facial expression generation. Based on the work by Waters [24], Terzopoulos et al. [21] further improved the facial expression generation process so that real-time simulation could be achieved by early graphics workstations. The simulation realism is also enhanced with the help of geometric and photometric information provided by active sensors. Transient expressions presented in a video could also be captured and analyzed through the technique of deformable contours (*snakes*). Different from the traditional physics-based or anatomy-based schemes, Zhang et al. [28] employed their own hierarchical structures consisting of four components: a *mass-spring* mesh for the facial skin, a layer of muscle actuators to control facial movement, a skull mesh, and other additional meshes to represent eyes and teeth. As a result, such a framework could

offer efficient and realistic simulation results. Pighin et al. [13] proposed to map a 2D face image onto a 3D surface model. In their approach, uncalibrated cameras are deployed to take photos of a person's face with a specific facial expression through different viewing angles. User's assistance may be required to calibrate the cameras' positions and orientations through the control points on the subject's face. A general 3D human head model is then deformed to match the observed control point information through interpolations. Such a process is repeated for different facial expressions. Once the correspondence is built, a morphing of 3D surface models is straightforward, allowing facial expressions to vary smoothly from interpolating facial textures. Liu et al. [10] improved the facial expressions by considering some subtle changes in facial expressions such as the illumination and appearance (e.g., creases and wrinkles) through a mechanism called *expression ratio image* (ERI). Together with a geometric warping, the generated results are more convincing. As sometimes it is difficult to obtain the ERI from the performer, Zhang et al. [27] proposed methods to improve the traditional *expression mapping* approach, where a subject's faces of neutral and a particular expressions, together with the positions of facial features, are first given, and then the same specific expression for a new person could be generated through a geometry-controlled image warping. An example-based approach is used to infer the necessary but possibly missing feature points from the given data, and details on expressions such as wrinkles could be simulated via this approach. Instead of using expert-coded facial parameters such as FACS, Wang et al. [22] proposed a HOSVD (*higher-order singular value decomposition*) scheme to learn the mapping between a particular person and his/her expressions from a given set of images consisting of different expressions, and then the corresponding expressions for a new or unknown person could be synthesized. Zhou et al. [29] proposed a *kernel-based factorization* model to parameterize facial expressions as well as facial identities. In particular, hybrid facial expressions could be synthesized by proper settings of related parameters.

Aiming for efficiently representing facial expressions and animations, MPEG4-SNHC (synthetic/natural hybrid coding), a sub-protocol of the MPEG4 standard for video compression, contains two components: *synthetic objects* and *natural objects*. One of the standards within the first component, when combined with a 3D human model, is to provide an efficient description for transferring the related parameter information regarding body motions and facial expressions in a real-time manner, thus increasing the associated compression ratio. These parameters can be divided into two categories: facial expression-related and body motion-related. The first category, pertaining to this work, consists of two parts: FDPs (facial definition parameters), which defines the normal and basic outlook of a face, and FAPs (facial animation parameters), which describes the dynamics of expressions. Raouzaiou et al. [15] made use of this scheme for modeling facial expression animations, and Fig. 1 shows their corresponding locations of the control points on a standard face (cited from [15]).

Based on the aforementioned classifications, our approach, to be discussed detailedly in Section 3, generates a 2D output image, and takes an image-based approach. This approach alters a facial expression through the relocations of control points around specific facial features, i.e., eyes, eyebrows, and mouth. Compared with existing approaches, where 3D head models (or together with other associated models) [11, 13, 28], a bunch of 2D reference images [10, 22, 27, 29], a group of facial

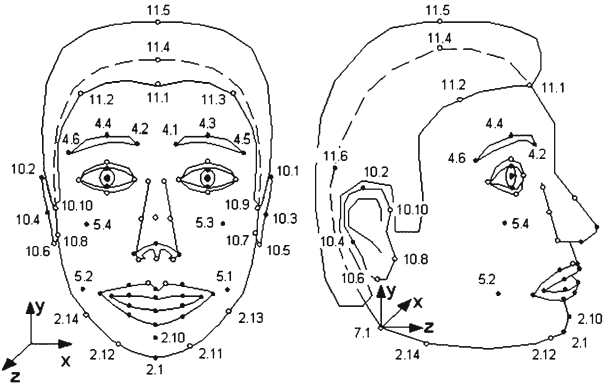


Fig. 1 The deployment of control points for facial expression animations in Raouzaiou et al.'s work

feature markers [13, 21, 27], a set of related parameters [2, 15, 24], and/or even non-trivial manual assistance [2, 13] are needed from the users, our approach requires nothing but a photo and few strokes. In addition, in terms of user assistance, ours, i.e., the drawing of three rough contours, is also comparatively easier than that of others' approaches where user involvement is deemed necessary. This simplicity, which we value most in this work, not only facilitates its usefulness to the general public, but also greatly reduces the involved complexity of generating desired expressions. However, we admit that such an approach achieves its efficiency at the expense of sacrificing some accuracy and expressiveness, as will be discussed in the last Section. Nevertheless, as a photo is the most common and oftentimes the only information that we could obtain, the functionality provided by our system could already be satisfactory to many users.

Moreover, as mentioned in Section 1, part of our contribution in this paper is to demonstrate how facial expressions could be used to interact with other multimedia elements. Along this line of research, Shugrina et al. [19] showed how facial expressions could affect the "mood" of an image by changing its colors accordingly. There is another interesting paper which is by far, the most similar paper to ours. Schubert [17] displayed the emotions of music by an artificial 2D cartoon-like face, called *Emotionface*, where the emotion analysis of the input music was done beforehand. Instead of using an artificial face for displaying music emotions, we show the emotions through a face with a more realistic look; and rather than analyzing the music manually and beforehand, we perform the analysis almost *on-the-fly* as the music plays.

3 Feature expression generation

The generation process of facial expression consists of six steps: thresholding, feature selection, feature refinement, micro-tuning, expression generation and image morphing. We further elaborate each of these steps in the ensuing sub-sections.

3.1 Thresholding

To facilitate feature identifications, as in a normal *edge detection* process, an input color image is usually *quantized* into a *binary image* through a *thresholding* proce-

dure. Such a thresholding process normally converts an input RGB image into a *grayscale* image first, where each pixel is only associated with an *intensity* value, and then each such intensity value is checked against a given *threshold*. A pixel with an intensity value greater than or equal to the threshold will be set to *white*, or *black* otherwise. Currently the intensity value of a RGB pixel is the sum of the values from the R, G, and B channels. Figure 2b is the result of running a thresholding process on Fig. 2a, the original input image, where the threshold is set to be 570. Note that as it is usually not possible to have a *universal threshold* which could always produce a satisfactory result for every image, we therefore allow users to dynamically adjust the setting of thresholds should he/she is not happy with the thresholded results. We have also tried to use other color systems such as YC_bC_r without noticing any significant improvement.

3.2 Feature selection

Given the fact that arbitrary images may come as inputs, and through numerous experiments, we have found that it is difficult to automatically detect the expression-

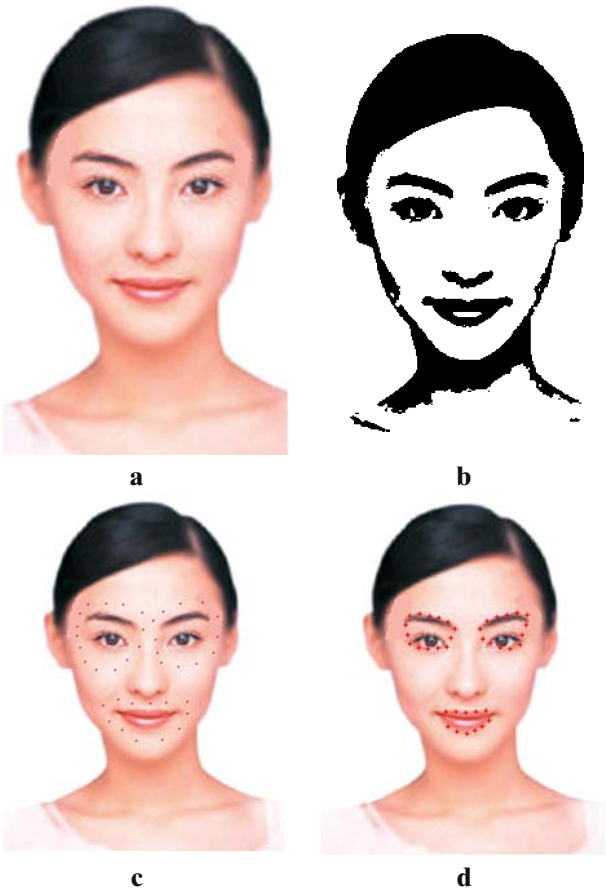


Fig. 2 **a** Original input photo. **b**, **c** and **d** are the results after thresholding, drawing contours, and applying the active contour algorithm, respectively

related facial features. For the sake of simplicity and convenience, we therefore require a user to draw three contours by marking control points around the following three facial features: left eye and eyebrow, right eye and eyebrow, and mouth, as shown in Fig. 2c. Alternatively we may apply the technique of *circle detection* to first identify the positions of eyeballs, and from where we could try to locate the exact contours of eyes, eyebrows, bridge of the nose, as well as the mouth. However, as the lighting on a subject's face may vary significantly, even a more intelligent approach as such still cannot always guarantee a success.

After the contour points are drawn, we apply the popular approach of *active contour*, or *snake* to shrink the initial contour to match more closely with our target facial features, as shown in Fig. 2d. Our current implementation of the snake algorithm follows from Williams et al.'s approach [25], which simplifies and speeds up the original version given by Kass et al. [8].

3.3 Feature refinement

Recall that it is for the reason of convenience that we require a user to mark only three feature regions. For having a more precise and detailed control of facial expression generation, after applying the aforementioned snake algorithm to closely locate the three feature regions, we need to further split an *eye-eyebrow* region into two separate regions, namely the *eye region* and *eyebrow region*, automatically. One way to do this is to track the eye-eyebrow regions with the technology of *level sets* [5, 18], which could support merging or splitting of contours, thus leading to the separation of the eye regions and eyebrow regions. However, as such an approach is more time consuming and still cannot always guarantee a perfect result, we resort to a simpler method by finding a route that could split the original region into two, as shown in Fig. 3a. Such a route apparently should intersect with neither the eye nor the eyebrow, and thus could be served as a splitting path. After such a route is located, we uniformly insert sample points along the route and each of the added sample points is duplicated for being used as a contour point for both the upper eyebrow region and the lower eye region, as shown in Fig. 3b, where points to be duplicated are marked in blue. Finally we apply the active contour algorithm again so that the contour points of both the eyebrow and eye regions could converge to two separate and tight feature contours, which are demonstrated in Fig. 3c.

In our current implementation, the way we locate the route is to first identify the *extremum points* of the involved eye-eyebrow region, as shown in Fig. 4, where K, KL, and KR denote the *topmost*, *leftmost* and *rightmost* points of the eyebrow region, respectively, while J, JL and JR the *bottommost*, *leftmost* and *rightmost* points of the eye region, respectively. The next step is to try to find a path that

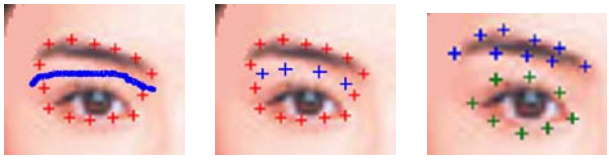


Fig. 3 The process of automatically splitting an eye-eyebrow region into two separate regions

connects the midpoint of KL and JL, and a point lying between KR and JR, without touching the eye and eyebrow regions. Figure 5 demonstrates such a process, where essentially we try to go rightwards as much as possible and at the same time avoid contacting facial features by going upwards or downwards until we reach the boundary delimited by KR and JR. Note that the images shown in this figure are just for demonstration purpose, and in our implementation, the *path-finding* process is in fact applied on top of the thresholded, black-and-white image shown in Fig. 2b. In other words, a path, starting from a white pixel, and moving rightwards, should go upwards or downwards without touching any black pixels until reaching the right delimitation.

Figure 6 demonstrates the final *five-region configuration* after the two groups of contour points of both eye-eyebrow regions converge with the proposed algorithm.

3.4 Micro-tuning

As input photos could vary significantly, the proposed algorithm may not always work well. As a final remedy, a user could inspect the previous five-region configuration and decide if he/she wants to fine tune the five resulting converged contours. If so, with one click, an editing window will pop up, in which each contour has already been converted into a *cardinal spline loop*, that is, a collection of smoothly joined spline curves which altogether form a loop. All the control points on the loop is *adjustable*, thus allowing the user to *micro-tune* the target contour. The corresponding *tension* coefficients are all set to zero, so that the defined curves are the smoothest. Figure 7 shows the facial features that are defined by five cardinal spline loops, which allow the user to fine tune the resulting facial feature contours generated from the previous phases.

3.5 Expression generation

After the boundaries of all desired facial features are properly represented by contour points, the next step is to generate target expressions.

3.5.1 JAFFE facial expression database

It should raise no objection that a facial expression could be represented by a collection of facial features while each of these features having a particular status. One could adopt the approach by Raouzaiou et al. [15], which in turn comes from Karpouzis et al.'s idea [7]. However, here we resort to a simpler model by making observations on the JAFFE (Japanese Female Facial Expression) database [6], shown in Fig. 8 (cited from [6]), and develop our own rules for facial expressions as follows. Figure 9 shows the facial model of this work. Basically we distinguish four emotions and make feature adjustments with respect to the basic neutral expression (no emotion). Note that each number in a parenthesis here represents a particular

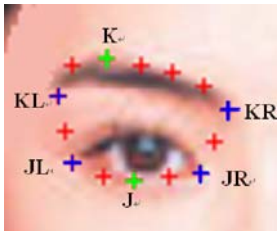


Fig. 4 The extremum points of an eye

Fig. 5 The process of automatically finding a path within an eye-eyebrow region

Fig. 6 The final five-region configuration after the refinement stage

Fig. 7 The cardinal spline loops of facial features

Fig. 8 The JAFFE databases

control point. To have a unified framework, each input photo is scaled or normalized to the resolution of 300 × 350.

- *Happy*

    1. (3)∼ (7), (11)∼(15): move upwards by 9 pixels.
    2. (1), (2), (8), (9), (10), (16): move upwards by 6 pixels.
    3. (22)∼(24), (30)∼(32): move upwards by 1 pixel.
    4. (34)∼(36): move upwards by 4 pixels.
    5. (33), (37): move upwards by 8 pixels.

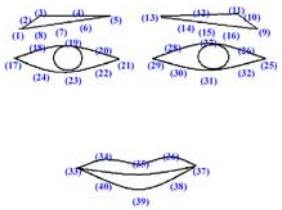


Fig. 9 The face model used in this work

- *Sad*

  1. (3)∼(7), (11)∼(15): the upward shift of a control point is proportional to the distance from this point to (3) (for the left eyebrow) and (11) (for the right eyebrow). Here the maximum shift is set to be 12 pixels.
  2. (17)∼(18): move leftwards and downwards by 2 pixels.
  3. (25)∼(26): move rightwards and downwards by 2 pixels.
  4. (33)∼(34), (40), (36)∼(38): the downward shift of a control point is proportional to the distance from this point to (35), and the maximum shift is set to be 6 pixels.

- *Angry*

  1. (1)∼(8): move rightwards by 5 pixels.
  2. (9)∼(13): move leftwards by 5 pixels.
  3. (4)∼(7): move rightwards by 3 pixels and the downward shift of a control point is proportional to the distance from this point to (3). The maximum shift is set to be 6 pixels.
  4. (12)∼(15): move leftwards by 3 pixels and the downward shift of a control point is proportional to the distance from this point to (11). The maximum shift is set to be 6 pixels.
  5. (18)∼(20), (26)∼(28): move downwards by 2 pixels.
  6. (22)∼(24), (30)∼(32): move upwards by 2 pixels.
  7. (33)∼(34), (40), (36)∼(38): the downward shift of a control point is proportional to the distance from this point to (35), and the maximum shift is set to be 5 pixels.

- *Surprised*

  1. (18)∼(20), (26)∼(28): move upwards by 2 pixels.
  2. (22)∼(24), (30)∼(32): move downwards by 2 pixels.
  3. (33)∼(34), (40), (36)∼(38): the downward shift of a control point is proportional to the distance from this point to (35), and the maximum shift is set to be 6 pixels.
  4. (2)∼(8), (10)∼(16): move upwards by 9 pixels.

3.5.2 Triangulation

As the change of an expression involves not only the contour, but also the interior, we therefore have to *triangulate* the five polygons defined by the five groups of contour points before and after the change of expressions. It can be shown that each of these five polygons is an *x-monotone polygon*, i.e., a polygon with the property that each vertical line could intersects with this polygon at most two points, as shown

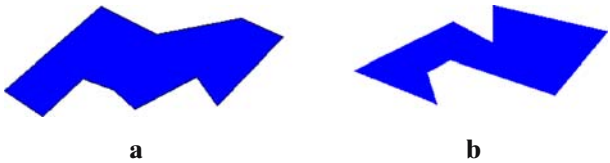
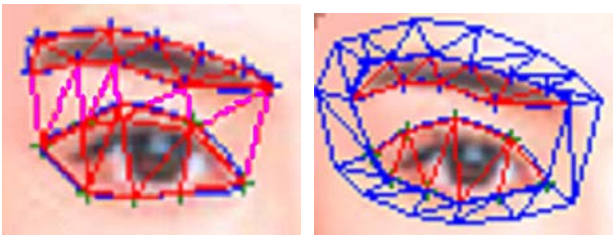


Fig. 10 **a** An x-monotone triangle. **b** A non-x-monotone triangle



a **b**

Fig. 11 **a** Triangulation between facial features. **b** Triangulation outside of facial features

in Fig. 10a, therefore a simple triangulation algorithm such as the one proposed by Fournier et al. [3], could be applied.

However, as sometimes it is not always possible to identify the exact contours of facial features, care must taken during the triangulation process for the regions close to the facial features, so that undesired distortions will not occur. For this purpose, we triangulate not only the area between facial features, as shown in Fig. 11a, but also the area slightly outside of the target region, as shown in Fig. 11b.

Figure 12 shows the final triangulations where areas between and outside facial features are included.

Once the triangulation of these polygons is done, the interior of each deformed region could then be determined using the *bilinear interpolation* [14] through the *barycentric coordinates* defined on all the triangles included in these polygons. This would always be feasible, as long as the change of feature contours does not lead to degeneracy, i.e., two control points clash into one, and it is indeed the case in our current implementation.

3.6 Image morphing

To have smooth transitions between different facial expressions, the images corresponding to different emotions are morphed through the techniques of *image morphing* [1, 20, 26], where we adopt particularly the *mesh warping* approach by Wolberg [26], as demonstrated by Fig. 13 (cited from [9]). In this Figure, the upper-left and lower-right images denote the source and target images, I_s, and I_t, respectively, while both of them have also been superimposed with meshes, M_s and

Fig. 12 The final triangulations

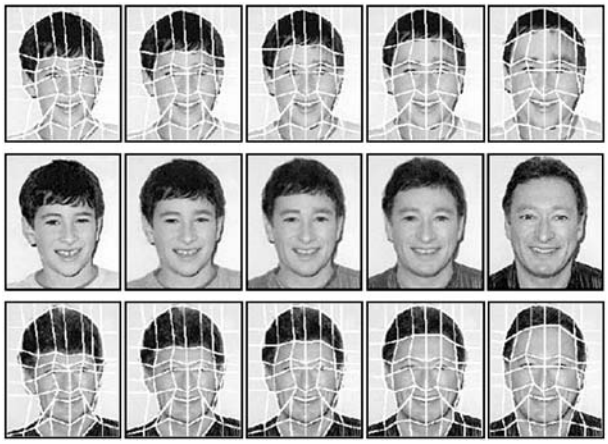


Fig. 13 A mesh warping morphing process

M_t, of the same *topology*, i.e., a 8×8 grid, respectively. Both meshes are *deformed* to fit facial features of the source and target images as much as possible. The procedure of mesh warping *linearly interpolates* the frames between I_s and I_t. For each intermediate frame, shown as the central image of a column (containing three rows of images) in Fig. 13, we perform the following four steps. First, the mesh, denoted by M, between M_s and M_t, is linearly interpolated. Second, I_s is *warped* into I_1, the top image of a column, using meshes M_s and M. Third, similarly, I_t is warped into I_2, the bottom image of a column, using meshes M_t and M. Finally, the resulting image (frame) is again linearly interpolated from I_1 and I_2. Rather than using quadrilateral meshes, our algorithm makes use of triangular meshes, as shown in Fig. 11, to represent facial features. However, the underlying interpolation principles are still the same.

Mesh warping is chosen for its capability of providing a better *local control* than other techniques, such as *field morphing* [1], and thus is more suitable for dealing with the subtle variation of expressions. Note that as long as the control points on the feature contours do not degenerate, there exists a natural correspondence between the control points on different expressions, and thus the success of mesh warping is guaranteed.

4 MIDI music analysis

As mentioned previously, in addition to generating facial expressions, another innovation of this paper is to show the possibility of using facial expressions to interact with other media. One exemplary application of such, though currently beyond the scope of this paper, is to integrate facial expressions with *speech emotion recognition* (in addition to general *speech recognition*) applied on top of the voice communication over modern MSN or Skype, where photos or *avatars* are used instead of real videos due to privacy concerns or bandwidth constraints. For instance, once an *angry* tone has been sensed from the vocal communication, a face with an angry look could then be synthesized at the other communication side, thus making a more vivid chatting experience.

Bearing a similar idea, and without plunging too deeply into the details of *speech processing*, we opt for music as the media to interact with facial expressions. Also as the processing of *MP3 or WAVE* files is more complicated, we further assume the input music to be in the format of MIDI, which is a music file format mainly used to communicate with specific instruments by representing a song via the description of pitch, length, strength (or velocity), type of instrument, and so on, for the constituent notes. To have music interact with the facial expressions, we first developed a tool for approximate music emotion analysis, and the algorithm of which will be detailed later. With this tool added in, our system could now interactively display the analyzed *music emotions* through the facial expressions generated by the aforementioned procedures. This integration could help to further prove the feasibility and efficiency of our facial expression generation system.

Before describing our music analysis algorithm, we begin by briefing what a MIDI file is composed of. A MIDI file, represented in a binary form, is composed of two kinds of *chunks*: a leading *header chunk*, labeled as *MThd*, and one or more *track chunks*, labeled as *MTrk*, that follow the header chunk. A header chunk describes the general information of a MIDI file, such as the type of MIDI, the number of track chunks, and the size of *delta-time*, which basically represents the smallest timing unit that could be specified for the ensuing *events* in this MIDI file. Here the type of MIDI refers to three possible types: one single track, multiple *synchronized* tracks, where all tracks must be played simultaneously, and multiple *independent* tracks, where all tracks could be played independently with each other. A track chunk is usually associated with a particular instrument, and consists of one or more pairs of *delta-time* and *event*. When the delta-time is set to 0, it means the current event is to be played simultaneously with the previous event. There are three kinds of MIDI events: *MIDI channel event*, *system exclusive event* and *meta event*. A MIDI channel event could be further classified as *note on*, *note off*, *controller*, *program change* events, and so on, to represent the cases where a note is initiated, a note is terminated, a control (pan, balance, soft pedal, a particular effect, etc.) is applied, and a change of instrument, respectively, together with the necessary parameter information, such as notes and velocity. A system exclusive event is normally used to provide the information regarding the software or hardware manufacturers. A meta event is mainly for annotation purpose. For example, the name of the instrument, copyright notice, lyrics, cue point, etc., could be recorded in this kind of events. Thus it is evident that we could collect all necessary information for music analysis from the MIDI channel events. To simplify, we assume the input MIDI to be composed of either a single track or synchronized tracks, so that we only need to deal with a single melody. As the detailed description of the MIDI specification should not be the focus of this paper, we refer interested readers to http://www.midi.org/ for further MIDI information.

In our system, we perform the MIDI analysis on a per *measure* basis, where a measure is a segment of time formed by a given number of *beats*. Currently our algorithm for distinguishing the emotion of a measure is shown in Fig. 14. Basically we first compare the number of notes within a measure against a threshold, which is set to be 7 in our current implementation after testing dozens of music pieces, to determine if the input measure enjoys a fast rhythm or not. If so, it will be classified as being *happy*. Otherwise, the average pitch of this measure is checked against the average pitch of the whole music. A measure with a lower averaged pitch is thought

Fig. 14 The algorithm for classifying the emotion of a given measure

```
Measure_Classifier(measure)
{
  Let N_notes = the number of notes in this measure
  Let Avg_measure = the average pitch of this measure
  Let Avg_whole = the average pitch of the whole music
  Let Avg_front_half = the average pitch of
      the front 50% highest pitches of the whole music

  if N_notes > Threshold then
    emotion = "happy"
  else if Avg_measure < Avg_whole then
    emotion = "sad"
  else if Avg_measure > Avg_front_half then
    emotion = "angry"
  else
    emotion = "happy"

  return emotion
}
```

to have a *sad* emotion, while a measure with a higher averaged pitch is further tested against the average pitch of the *front 50% highest pitches of the whole music*. A measure with a higher averaged pitch is considered *angry*, or *happy* otherwise. Note that according to this algorithm, a measure could be classified as being *happy* in two cases.

As far as the classification is concerned, there are at least two shortcomings with this algorithm. First, it is evident that the emotions of *surprised* and *neutral* are missing from this classification scheme. The missing of the *surprised* emotion is a very unfortunate decision, as so far we still cannot find a satisfactory rule to distinguish it from the *angry* emotion through numerous experiments. The missing of the *neutral* emotion, on the other hand, is relatively less serious, and in fact, a little bit deliberate, as the *neutral* expression will appear anyway during the interaction process. This is mainly due to our current implementation for morphing between expressions; that is, to have a smoother look of the transition, the *neutral* expression will always be inserted between two different expressions before the morphing process is applied to interpolate the intermediate frames. Second, the use of pitch average may not always

Fig. 15 The system interface for interactively generating facial expressions

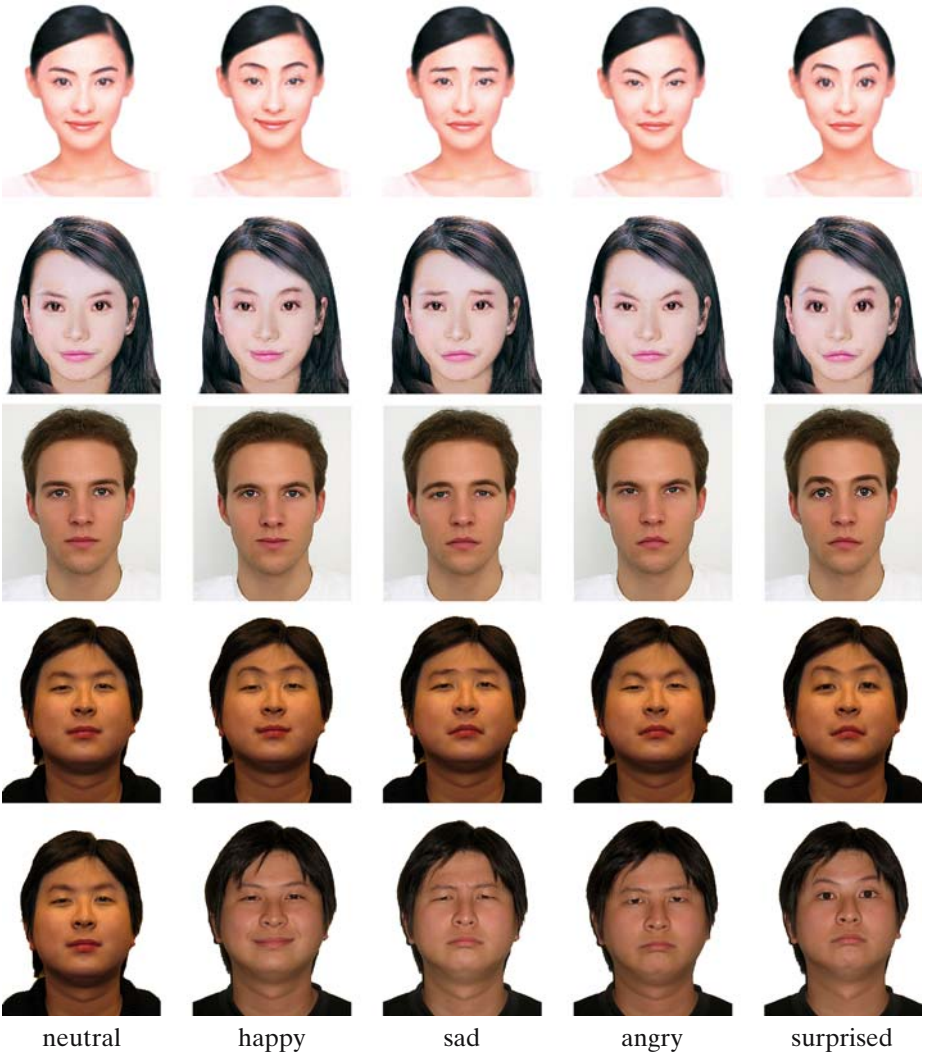


Fig. 16 The facial expression results of four persons. The synthetic results are shown in the *first four rows* while the real expressions of the fourth person are shown in the *last row* for comparisons

be precise. For example, consider that case where a very high pitch may coexist with several low pitches within one measure, thus resulting a *happy* classification, which might not be correct. However, in practice, such ambiguity does not happen very often.

As a further remark, the reasons that we resort to such a heuristic classification scheme are the following. First, a comprehensive and accurate music emotion analysis scheme is still an ongoing research area [16, 23]. Second, as mentioned in Section 1, we want our system to be simple and fast, and thus it could demonstrate a more responsive interaction between MIDI music and facial expressions. Note that the preprocessing work for these rules involves only one linear scan of all the notes,

and then the average pitch of the whole music and of the 50% highest pitches could be easily and quickly computed. Also note that the involved sorting process could be done by a linear-time algorithm, such as *counting sort*, as all the pitches could only come from a finite set of values, and thus could be treated as integers. At the run time, each measure could thus be classified *on the fly* in a nearly real-time fashion. It should also be self-evident that our music analysis scheme does not only apply to MIDI, but also to other formats of music as well, such as WAVE and MP3, as long as the same necessary music information could be extracted and processed beforehand.

5 Results

We have conducted our experiments on a Pentium IV 3.0GHz machine with 512MBytes memory, running on the Windows XP operating system. Figure 15 shows the interface that we use to perform facial expression generation, where various parameters could be tuned to achieve the best results.

Figure 16 demonstrates the results generated by our system. Based on the input photos of four persons, shown in the leftmost column, we have synthesized four expressions: *happy*, *sad*, *angry* and *surprised*.

As shown in this figure, when compared with the real expressions, our generated results do exhibit noticeable difference, especially for the third and fourth persons. For example, the third person's double-eyelid eyes may complicate the eye tracking process, thus leading to less precise results. Another example is the difference between having wrinkles or not, as shown in the *sad* expression for the fourth person. It should be apparent that the effect of wrinkles is not addressed so far by our proposed deformation and interpolation scheme. One more non-negligible difference is the enlarged eye-size, as shown in the *surprised* expression for the fourth person. This is due to a more exaggerated expression made by the fourth person during the experiments. Nevertheless, here we would like to make the following two remarks. First, *all the results are generated solely from the formulas presented in Section* 3.5.1. Second, there should be nearly no doubt that, *when the synthesized images are displayed alone, the desired expressions could be observed*. These are the reasons why our proposed simple and fast algorithm could claim its contribution.

Based on these synthesized images, we could "pre-generate" animations of image morphing between expressions, so that at the run time, specific animations could be applied when corresponding music emotions are detected. Figure 17 shows the

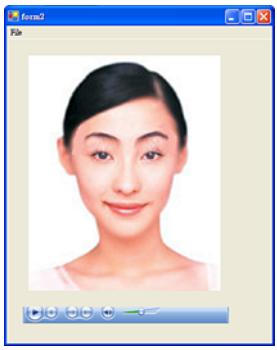


Fig. 17 The interface displaying analyzed music emotions through facial expressions

Table 1 The information of the 5 MIDI files

MIDI	MIDI leng. (s)	Proc. time (s)	Emotions
a.mid	68	10	Happy, sad
b.mid	71	11	Happy, sad, angry
c.mid	132	13	Happy, sad
d.mid	152	15	Happy, sad
e.mid	154	15	Happy, sad, angry

interface where our synthesized expressions interact on the fly with the analyzed music emotions.

To interact with music, five MIDI files are chosen from a wide variety of pop music from Taiwan and Hongkong, based on the assumption that the selected music should be able to arouse different kinds of feelings when listened, and the corresponding information is shown in the first three columns of Table 1. As shown in the table, the lengths of the MIDI music pieces range from 68 to 154 seconds, and for all the input MIDI music, the system processing time is never more than 15 seconds, which is mainly spent on interpolating the intermediate frames (15 frames per second) for morphing between different expressions. The last column of this table lists the detected emotions from these MIDI files, using the algorithm shown in Fig. 14. Note that, as mentioned in the previous section , we do not distinguish the *surprised* emotion from the MIDI files. Also as said earlier, the image size is normalized to be 300×350, therefore larger images would require proportionally larger amount of processing time. The total end-to-end time, that is, from the time an input photo is given, till the time different expressions are generated, is about 1 min.

6 Conclusions and future work

We have proposed and implemented a facial expression generation system that provides a user-friendly interface for users to input desired photos. Together with few simple strokes to roughly identify facial features, our system could then quickly generate different vivid facial expressions using the designated face, while the whole interactive process normally requires only about 1 min. To prove the effectiveness of our facial expression generation system, we further integrate it with a primitive capability for music emotion analysis; that is, we hope the emotions detected from a MIDI music piece could be reflected by our generated facial expressions. Experimental results show that our system, with negligible amount of preprocessing time, could interact with a given MIDI music piece in a real-time fashion. We believe that such a simple and efficient approach could find its use in the modern world. For example, *it could be used as a companion tool in MSN or Skype, or on some hand-held devices such as PDAs or smart phones* for efficiently and vividly displaying personal facial expressions, when the underlying bandwidth is constrained.

In the future, we plan to automate the whole process completely even though the involved manual effort in our current implementation is already acceptable. This would definitely pose more challenges as personal photos may present huge varieties or imperfection. For example, a man's hair may block his eye/eyebrow partially or completely, thus leading to the difficulty of properly distinguishing the facial features. One more interesting extension is to generate corresponding expressions

when a subject's mouth is opened (e.g. for the expressions of wild laughing or astonishing). Another possibility is to add wrinkles, as shown in Liu et al.'s work [10]. Additionally, expressions may have personal styles. For instance, person *A*'s angry face could be quite different from that of person *B*, therefore we could also take *personalization* into account for generating more realistic results. Furthermore, we would like to generalize this scheme to a 3D model so that more general facial expression changes or movements could be represented. Most importantly, the above generalization should be simple, fast and efficient. Finally, the music analysis part, mainly serving for a demonstration purpose now, could definitely be improved as it is admittedly very primitive at this present. For example, in addition to pitches and tempos, the involved instruments may also play an important role on distinguishing the associated emotions.

References

1. Beier T (1992) Feature-based image metamorphosis. In: SIGGRAPH '92, pp 35–42
2. Ekman P, Friesen W (1978) Facial action coding system: a technique for the measurement of facial movement. Consulting Psychologists Press, Palo Alto, CA
3. Fournier A, Montuno DY (1984) Triangulating simple polygons and equivalent problems. ACM Trans Graphics 3:153–174
4. Gleicher M (1998) Retargetting motion to new characters. In: SIGGRAPH '98, pp 33–42
5. Han X, Xu C, Prince JL (2003) A topology preserving level set method for geometric deformable models. IEEE Trans Pattern Anal Mach Intell 25(6):755–768
6. Lyons M, Akamatsu S, Kamachi M, Gyoba J (1998) Coding facial expressions with Gabor wavelets. In: Proceedings of Third IEEE International Conference on Automatic Face and Gesture Recognition. Nara Japan, IEEE Computer Society, pp 200–205
7. Karpouzis K, Tsapatsoulis N, Kollias S (2000) Moving to continuous facial expression space using the MPEG-4 facial definition parameter(FDP) set. In: SPIE Electronic Imaging 2000, pp 443–450
8. Kass M, Witkin A, Terzopoulos D (1988) Snakes: active contour models. Int J Comput Vision 1:321–332
9. Lischinski D (2006) Selected topics in computer graphics course slides. http://www.cs.huji.ac.il/~danix/
10. Liu Z, Shan Y, Zhang Z (2001) Expressive expression mapping with ratio images. In: SIGGRAPH '01, pp 271–276
11. Parke FI (1972) Computer generated animation of faces. In: Proceedings of Annual ACM Conference
12. Parke FI, Waters K (1996) Computer facial animation. AK Peters, Wellesley, MA
13. Pighin F, Hecker J, Lischinski D, Szeliski R, Salesin DH (1998) Synthesizing realistic facial expressions from photographs. In: SIGGRAPH '98, pp 75–84
14. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1997) Numerical recipes in C, 2nd edn. Cambridge Univ. Press
15. Raouzaiou A, Tsapatsoulis N, Karpouzis K, Kollias S (2002) Parameterized facial expression synthesis based on MPEG-4. In: EURASIP '02, pp 1021–1038
16. Schubert E (1999) Measurement and time series analysis of emotion in music. PhD thesis, University of New South Wales
17. Schubert E (2004) Emotionface: prototype facial expression display of emotion in music. In: Proceedings of ICAD-04
18. Sethian A (1999) Level set methods and fast marching methods, 2nd edn. Cambridge Univ. Press, Cambridge, UK
19. Shugrina M, Betke M, Collomosse J (2006) Empathic painting: interactive stylization through observed emotional state. In: NPAR '2006, pp 87–96
20. Smythe DB (1990) A two-pass mesh warping algorithm for object transformation and image interpolation. Technical Report ILM Technical Memo No. 1030, Computer Graphics Department, Lucasfilm Ltd
21. Terzopoulos D, Waters K (1993) Analysis and synthesis of facial image sequences using physical and anatomical models. IEEE Trans Pattern Anal Mach Intell 15(6):569–579

22. Wang H, Ahuja N (2003) Facial expression decomposition. In: Proceedings of Nineth IEEE Int Conference on Comput Vision, pp 958–965
23. Wang M, Zhang N, Zhu H (2004) User-adaptive music emotion reconition. In: Proceedings of the ICSP'04, pp 1352–1355
24. Waters K (1987) A muscle model for animating three-dimensional facial expression. In: SIGGRAPH '87, pp 17–24
25. Williams D, Shah M (1990) A fast algorithm for active contours. In: The 3rd IEEE Int Conference on Comput Vision, pp 592–595
26. Wolberg G (1990) Digital image warping. IEEE Society Press, Los Alamitos
27. Zhang Q, Liu Z, Guo B, Terzopoulos D, Shum H (2006) Geometry-driven photorealistic facial expression synthesis. IEEE Trans Vis Comput Graph 12(1):48–60
28. Zhang Q, Prakash EC, Sung E (2003) Efficient modeling of an anatomy-based face and fast 3D facial expression synthesis. Comput Graph Forum 22(2):159–169
29. Zhou C, Lin X (2005) Facial expressional image synthesis controlled by emotional parameters. Pattern Recogn Letters 26(16):2611–2627

Chuan-Kai Yang received his Ph.D. degree in computer science from Stony Brook University, USA, in 2002, and his M.S. and B.S. degree in Computer Science and in Mathematics from National Taiwan University in 1993 and 1991, respectively. He has been an Assistant Processor of the Information Management Department, National Taiwan University of Science and Technology since 2002. His research interests include computer graphics, scientific visualization, multimedia systems, and computational geometry.

Wei-Ting Chiang received his Bachelor's degree in information Management at National Cheng Kung University in Taiwan in 2004, and his Master degree in Information Management at National Taiwan University of Science and Technology in 2006, respectively. His research interest is on multimedia processing. He is now working in the industry.