Interactive facial expression editing based on spatio-temporal coherency

Jing Chi¹ · Shanshan Gao¹ · Caiming Zhang¹

Published online: 4 May 2017

© The Author(s) 2017. This article is an open access publication

Abstract We present a novel approach for interactively and intuitively editing 3D facial animation in this paper. It determines a new expression by combining the user-specified constraints with the priors contained in a pre-recorded facial expression set, which effectively overcomes the generation of an unnatural expression caused by only user-constraints. The approach is based on the framework of example-based linear interpolation. It adaptively segments the face model into soft regions based on user-interaction. In dependently modeling each region, we propose a new function to estimate the blending weight of each example that matches the user-constraints as well as the spatio-temporal properties of the face set. In blending the regions into a single expression, we present a new criterion that fully exploits the spatial proximity and the spatio-temporal motion consistency over the face set to measure the coherency between vertices and use the coherency to reasonably propagate the influence of each region to the entire face model. Experiments show that our approach, even with inappropriate user's edits, can create a natural expression that optimally satisfies the user-desired goal.

Keywords Facial animation · Editing · Spatio-temporal motion · Adaptive segmentation

1 Introduction

Providing users interactive and efficient editing tools for producing expressive and realistic facial animations is a chal-

lenging problem in computer animation. The tool should be intuitive and easy to use. It allows users to select simple control elements, e.g., points, curves on a 3D face model and simply edit them to create new expressions. Besides, the tool should produce nature and convincing facial animations. However, developing such a tool is difficult because the editing information controlled by user is so low-dimensional compared to the model with thousands of degrees of freedom. Thus, the user's edits cannot be used to fully determine an expression since it may result in an unnatural expression. Moreover, in some cases, the user's control is not appropriate due to lack of experience or some other reasons, which also leads to unnatural expressions.

Motivated by the above questions, in this paper we present a novel approach to facial animation editing. Our approach allows the user to interactively pick pixels on 2D screen and intuitively displace them to change expressions of a 3D face model. Our basic idea is to represent the new expression as a linear combination of the pre-recorded examples. The main contributions include: (1) instead of only depending on the user-edited constraints, we integrate the pre-recorded facial expression data as a priori into our approach to jointly determine the new expression. It effectively overcomes the unnatural expressions generated only by the user-defined constraints. (2) We introduce a new function to estimate the blending weights of face examples to represent the deformation of control point. The function considers not only the optimal match to the user's constraints, but also the likelihood between face examples and the user-desired expression. Minimizing the function can get reasonable blending weights and establish accurate deformation model. (3) We propose a new criterion that exploits the spatio-temporal motion consistency over the whole face set to measure the coherency of the model vertices with respect to the control points. For a control point, the larger the coherency between it and the

✉ Jing Chi
peace_world_cj@hotmail.com

¹ Department of Computer Science and Technology, Shandong University of Finance and Economics, ji'nan, China

vertex, the greater influence are imposed on the vertex. This criterion establishes an influence map for each control point, and reasonably propagates the influence of each control point to the entire face model.

2 Related work

Early example-based facial expression editing work [1–3] often created a new expression by linearly interpolating the pre-recorded face examples in the original space. They estimated the weight of each example from the user-defined constraints and then blend the examples. These methods are quick and easy to implement, but show limitations when producing realistic expressions. Only using one or just a few constraints specified by user is difficult to create a reasonable and natural expression since the problem is underconstrained.

Several example-based work used principle component analysis (PCA) to solve the underconstrained problem. Blanz and Vetter [4] and Chai et al. [5] used PCA to compute the blending weights that maximize their likelihood with respect to the pre-recorded face data. Lau et al. [6] combined the user's inputs with a priori learned from the captured face data in the reduced PCA subspace to edit expression. Lewis and Anjyo [7] and Seo et al. [8] used PCA to automatically create a 3D space where each face example corresponds to a position, and directly manipulated these positions to edit expressions. Cetinaslan et al. [9] directly manipulated the face model by simply sketching on it. PCA effectively reduces the data dimensionality, but it lacks physical meaning and semantic interpretation for expression. Using PCA generally needs to segment the face model into separate regions to get good results.

Many editing techniques [10–14] that focus on segmenting the face model have been developed. Ma et al. [15] divided the model into six disjoint regions using the physically-motivated segmentation framework [10], and edited each region with the region-based ICA scheme. Tena et al. [16] segmented the model into multiple PCA sub-models sharing boundaries, and created a new expression by user's constraints and boundary consistency. Rigid segmentation in these methods decouples the natural correlation between different parts of a face, but in practice, the segmentation depends on what expression the user desires.

Zhang et al. [17] proposed local influence map to overcome the under- or over-segmenting problem. It segments the face model into soft regions based on user-specified control points, models each region and blends them by defining an influence map for each control point. Inspired by [17], we introduce influence maps into our approach to guide the adaptive soft region segmentation and region blending. But we use the spatial proximity and spatio-temporal motion consistency embedded in the priori face set to build the influence

Fig. 1 The pipeline of our approach

map, which leads to a more accurate result compared to [17] that used normalized radial basis functions. Moreover, we extend our approach by combining expression cloning technique, overcoming the limitation in [17] that only face model having the priori can be edited.

Recently, spacetime editing methods [15, 18–21] that can propagate the editing on a single face frame across the whole sequence have been explored. For example, [19] built a Poisson equation to propagate user's modifications at any frames to the entire sequence.

3 Our approach

Figure 1 illustrates the pipeline of our approach. It consists of four steps: (1) *User editing* given a default face model, users can edit the model by selecting some individual vertices (we call them control points) and imposing constraints on them. The point constraints are specified in 2D screen space. The interface allows users to interactively pick and drag points on screen until a desired expression is created. (2) *Modeling the deformations of control points* for each control point, we represent its deformation caused by user's editing as a linear combination of the examples in the pre-recorded facial expression set. We estimate the weight of each example based on its proximity to the user-desired expression and the user-edited constraints. (3) *Soft region segmentation and influence map establishment* for each control point, we compute the coherency of each model vertex with respect to it. We depend on the coherency values to adaptively segment the face model into different soft regions, each containing a control point, and to establish an influence map for each control point. (4) *Soft region blending* we blend the soft regions into a single expression based on the influence map of each control point. The influence of each region will decrease as it spreads over the entire face model.

3.1 User editing

Our approach starts with a 3D face model in the pre-recorded set that has a neutral expression. Our point constraints allow users to select any individual vertices on the 3D face model and change their positions to edit expressions interactively.

In order to provide an intuitive and convenient interactive interface, our approach allows users to specify the point constraints in 2D screen space. Specifically, the user can select each 3D vertex by picking a pixel on the 2D screen, and change its 3D position by dragging its corresponding 2D pixel to the target pixel position. Suppose the user chooses L source pixels on the screen, given as $\{s_l | l = 1, \dots, L\}$, and specify their target positions at pixel $\{p_l | l = 1, \dots, L\}$. We first perform ray tracing with the source pixels to select the 3D control points $\{v_l | l = 1, \dots, L\}$ on the face model. Then, our task is to create a new expression mesh on which each selected 3D control point v_l , at its new position q_l , projects onto its corresponding 2D target position p_l in the current camera view.

We give the mapping relationship between the 3D coordinate and the 2D projection of a vertex of face model. Let q denote the 3D coordinates and p denote the 2D projection, i.e., the 2D pixel coordinates, get

$$\begin{pmatrix} \omega p \\ \omega \end{pmatrix} = \begin{pmatrix} c_1 & c_2 & c_3 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} r_1^T & t_1 \\ r_2^T & t_2 \\ r_3^T & t_3 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} q \\ 1 \end{pmatrix} \quad (1)$$

where r_i^T is the row vector of the camera rotation matrix, t_i is the component of the camera translation vector, and f is the focal length of the camera. c_i refers to the intrinsic camera parameters. $c_1 = [\frac{s_w}{2} \ 0]^T$, $c_2 = [0 \ -\frac{s_h}{2}]^T$, $c_3 = [\frac{s_w}{2} \ \frac{s_h}{2}]^T$. Here s_w and s_h is respectively the width and height of the 2D screen.

From Eq. (1), we obtain the nonlinear function about 3D coordinates q of a vertex and its 2D projection p as

$$p = F(q) = \begin{pmatrix} \left(\frac{f r_1^T q + f t_1}{r_3^T q + t_3} + 1 \right) \cdot \frac{s_w}{2} \\ \left(1 - \frac{f r_2^T q + f t_2}{r_3^T q + t_3} \right) \cdot \frac{s_h}{2} \end{pmatrix} \quad (2)$$

Based on Eq. (2), our approach enables the users to control the pixels on 2D screen until get the final expression they desired.

3.2 Modeling deformations of control points

We independently model the deformation of each 3D control point which is specified by the user's editing. We represent the deformation as a linear combination of examples in the pre-recorded face set. The priori embedded in the natural facial expression set is used to ensure the creation of a natural expression. We introduce a new metric for optimizing the blending weight of each example. The metric not only

satisfies the user-edited constraints, but also consider the proximity between the example and the user-desired expression.

The input face set in our approach can be a spacetime mesh sequence reconstructed from the time-varying point clouds, each mesh with the same topology, or a sequence of meshes with different expressions obtained by editing the vertex positions of a neutral face model, each mesh also has the same topology. Supposing that the input sequence is made up of M frames, each with N vertices. At frame m , the mesh is given as $T_m = \{v_{i,m}\}, i = 1, \dots, N$. Without loss of generality, let the first frame T_1 be the face model with a neutral expression. For convenience, let $\{v_l | l = 1, \dots, L\}$ denote the user-specified 3D control points on the face model, and q_l denote the 3D target position where v_l should be at. Then for each control point v_l , its deformation can be expressed as the following linear combination

$$q_l = w_1 v_{l,1} + w_2 v_{l,2} + \dots + w_M v_{l,M} \quad (3)$$

where w_m is the blending weight of the input mesh T_m , $m = 1, \dots, M$.

The choice of weight w_m , $m = 1, \dots, M$ needs to consider the proximity of each mesh T_m to the user-specified constraint. The higher the proximity is, the greater the contribution to the user-desired result is, i.e., the greater the weight is. We use the space distance between the control point and its corresponding vertex on the input mesh to measure the proximity. Closer distance means greater possibility that the user will edit the face model into the expression of the input mesh, which indicates that the input mesh has higher proximity with the user's desired expression. Specifically, for each control point v_l , we estimate the blending weight of each input mesh by minimizing

$$E'(\mathbf{W}) = \sum_{m=1}^M \left(1 + \|v_{l,m} - q_l\|^2 \right) w_m^2 \quad (4)$$

where $\mathbf{W} = [w_1 \ w_2 \ \dots \ w_M]^T$, and $\|\cdot\|$ is the Euclidean distance. Obviously, small weights are encouraged for far input meshes when minimizing Eq. (4). Addend 1 is used to avoid neglect of the items in which the Euclidean distance is too small.

Combining Eqs. (3) and (4), we get the metric of computing the blending weights for control point v_l as

$$E(\mathbf{W}) = \theta_1 \left\| \sum_{m=1}^M w_m v_{l,m} - q_l \right\|^2 + \theta_2 \sum_{m=1}^M \left(1 + \|v_{l,m} - q_l\|^2 \right) w_m^2 \quad (5)$$

where θ_1 and θ_2 are used to blend two constraint terms.

The metric in Eq. (5) is specified in 3D. As discussed in Sect. 3.1, our approach allows the user to control the projection on 2D screen. Supposing that the target projection positions specified by the user are $\{\mathbf{p}_l | l = 1, \dots, L\}$, we modify Eq. (5) as follows

$$E(\mathbf{W}) = \theta_1 \left\| F \left(\sum_{m=1}^M w_m \mathbf{v}_{l,m} \right) - \mathbf{p}_l \right\|^2 + \theta_2 \sum_{m=1}^M \left(1 + \left\| F(\mathbf{v}_{l,m}) - \mathbf{p}_l \right\|^2 \right) w_m^2 \quad (6)$$

where $F(\cdot)$ is defined in Eq. (2). We use $F(\cdot)$ to project the new 3D position of v_l , computed by linear combination of the input meshes, onto the 2D screen to get its 2D position. Obviously, the position should be as close to the user-specified position $\mathbf{p}_l$ as possible. Meanwhile, we use $F(\cdot)$ to project $v_{l,m}$ onto its 2D position, and use the distance between the 2D position and the target pixel $\mathbf{p}_l$ to measure the proximity of mesh T_m to the user-specified expression. We minimize Eq. (6) using L-BFGS-B, a fast quasi-Newtonian method. We set $\theta_1 = 2$ and $\theta_2 = 1$ in our experiments. For each control point v_l , we optimize Eq. (6) to get a set of weights, represented as $\mathbf{W}_l$, that are used to linearly blend the input meshes to represent the deformation of v_l .

3.3 Influence maps and region segmentation

According to the control points specified by user, we adaptively segment the face model into different soft regions, each has a control point. The adaptive segmentation in runtime can avoid the under- or over-segmentation problem existed in many traditional methods. The soft regions are allowed to partially overlap each other, that is, a vertex can be classified into different regions. It is reasonable because the motion of a vertex may be influenced by multiple control points. In each soft region, the influence that the control point imposes on different vertices are different. Specifically, greater influence is imposed on the vertices that have close correlation with the control point. We therefore define a local influence map for each control point to reflect the variation of its influence. The soft region segmentation and influence map establishment are both based on the coherency between the control points and the vertices. We propose a new criterion (We call it the coherency criterion) to measure the coherency that fully exploits the spatio-temporal motion consistency of vertices over the entire input sequence. Using the coherency criterion, we can compute the coherency value of each vertex with respect to each control point.

The coherency criterion To analyze the coherency between a vertex and a control point on a face model, we consider not

Fig. 2 Coherency of $v_{i,1}$ and v_l on frame T_m

only their spatial proximity on the face model, but also the motion consistency between them over the whole sequence along the time axis. That is, if the vertex and the control point have small spatial distance and large spatio-temporal movement consistency, then they tend to have high coherency, and it means that they will undergo the similar deformations in generation of a new expression. Given a vertex $v_{i,1}$ and a control point v_l on the face model T_1 , we first respectively measure their coherency in each frame T_m of the input face mesh sequence as follows

$$b_{i,l}^m(v_{i,1}, v_l) = d_{i,l}^m(v_{i,1}, v_l) \cdot r_{i,l}^m(v_{i,1}, v_l) \cdot s_{i,l}^m(v_{i,1}, v_l), \\ m = 2, \dots, M, i = 1, \dots, N, l = 1, \dots, L \\ d_{i,l}^m(v_{i,1}, v_l) = 1 - \frac{\|v_{i,1} - v_l\|}{\max_{i,j} \{\|v_{i,1} - v_{j,1}\|\}}, \\ r_{i,l}^m(v_{i,1}, v_l) = \frac{1}{2} \left(1 + \frac{(v_{i,m} - v_{i,1}) \cdot (v_{l,m} - v_l)}{\|v_{i,m} - v_{i,1}\| \cdot \|v_{l,m} - v_l\|} \right), \\ s_{i,l}^m(v_{i,1}, v_l) = \min \left\{ \frac{\|v_{i,m} - v_{i,1}\|}{\|v_{l,m} - v_l\|}, \frac{\|v_{l,m} - v_l\|}{\|v_{i,m} - v_{i,1}\|} \right\}. \quad (7)$$

where M is the number of frames in input sequence, N is the number of vertices in each frame, L is the number of control points. The values of $d_{i,l}^m(v_{i,1}, v_l)$, $r_{i,l}^m(v_{i,1}, v_l)$ and $s_{i,l}^m(v_{i,1}, v_l)$ range from 0 to 1, respectively.

As shown in Fig. 2, $d_{i,l}^m(v_{i,1}, v_l)$ measures the spatial proximity between $v_{i,1}$ and v_l by using the Euclidean distance between them, and $d_{i,l}^m(v_{i,1}, v_l)$ gets larger as they are close to each other. This term shows the intuition that physically close vertices have high coherency. $r_{i,l}^m(v_{i,1}, v_l)$ estimates the approximation of directions that $v_{i,1}$ and v_l move across the input sequence by measuring the angle between their motion vectors. $r_{i,l}^m(v_{i,1}, v_l)$ gets larger as they move in the more similar directions. $s_{i,l}^m(v_{i,1}, v_l)$ measures the similarity of the moving speeds of $v_{i,1}$ and v_l by comparing the ratio of their movement distances from the first frame to the current frame. $s_{i,l}^m(v_{i,1}, v_l)$ gets larger as they move with the more approximative speed ratio. The two terms $r_{i,l}^m(v_{i,1}, v_l)$ and $s_{i,l}^m(v_{i,1}, v_l)$ show the intuition that the vertices that have the more spatio-temporal motion consistency are more likely to have high coherency. The three terms all achieve their maximum values when vertex $v_{i,1}$ is coincident with control point v_l , in this case, their distance is zero, and their movement is

perfectly coherent. Additionally, $r_{i,l}^m(v_{i,1}, v_l)$ and $s_{i,l}^m(v_{i,1}, v_l)$ are both set to 1 if any of the two motion vectors $v_{i,m} - v_{i,1}$ and $v_{l,m} - v_l$ is a zero vector. In this case, it is hard to judge the movement consistency of two vertices if one or both of them have no motion, so the coherency is determined only by $d_{i,l}^m(v_{i,1}, v_l)$.

With Eq. (7), we get the coherency value $b_{i,l}^m(v_{i,1}, v_l)$ of vertex $v_{i,1}$ with respect to control point v_l at each frame T_m . We average all the coherency values as

$$b_{i,l}(v_{i,1}, v_l) = \frac{1}{M-1} \sum_{m=2}^M b_{i,l}^m(v_{i,1}, v_l), \quad (8)$$

$$i = 1, \dots, N, \quad l = 1, \dots, L$$

to get the final coherency coefficient of $v_{i,1}$ respect to v_l .

Soft region segmentation According to the coherency coefficients computed with Eq. (8), the vertices of the face model are automatically classified into L soft regions, each containing a control point. A vertex can be classified into two or more soft regions since it may have correlations with multiple control points. Thus, these soft regions are partly overlapped.

Influence maps establishment In each soft region, the deformation of each vertex is driven by the control point in the region. Obviously, the degree of each vertex that is affected by the control point is different. The degree depends on the coherency between the vertex and the control point. Therefore, for each control point, we define a local influence map based on its coherency with each vertex computed using Eq. (8). The larger the coherency is, the greater the influence is. Specifically, in the region containing control point v_l , the deformation of a vertex $v_{i,1}$ can be expressed as

$$q_{i,1}^l = b_{i,l}(v_{i,1}, v_l) \cdot \mathbf{W}_l \cdot (\mathbf{v}_{l,1} \ \mathbf{v}_{l,2} \ \dots \ \mathbf{v}_{l,M})^T \quad (9)$$

where $\mathbf{W}_l$ is the blending weights set that has been computed with Eq. (6) for modeling the deformation of control point v_l . $q_{i,1}^l$ is the new position of $v_{i,1}$, it is also represented as a linear combination of the input meshes, but its blending weights depend on both its coherency with v_l and the blending weights set of v_l . Equation (9) models the deformations of each soft region.

3.4 Soft region blending

We blend into the soft regions to propagate the influence of each region over the entire mesh to generate a final single expression. The blending is guided by the local influence map of the control point in each region, and it means that, for each region, its influence will decrease with its spreading over the entire face mesh. Using region blending, we can get a large number of expressions that do not exist in the input sequence.

Specifically, for each of the L soft regions, a blending weights set $\mathbf{W}_l$ for its control point has been computed by optimizing Eq. (6). Then, for a vertex $v_{i,1}$ of the face model, its final deformation will be influenced by multiple control points since it may locate in two or more regions. In each region to which it belongs, its new position $\mathbf{q}_{i,1}^l$ resulted by the deformation of the control point can be computed by using Eq. (9), so its final position that is used to create the new expression should be represented as the linear combination of its new positions in each region. The final deformation of a vertex $v_{i,1}$, $i = 1, \dots, N$ can be expressed as follows

$$\mathbf{q}_{i,1} = \mathbf{W}_{i,1} \cdot (\mathbf{v}_{l,1} \ \mathbf{v}_{l,2} \ \dots \ \mathbf{v}_{l,M})^T$$

$$\mathbf{W}_{i,1} = \sum_{l=1}^L B_{i,l}(v_{i,1}, v_l) \cdot \mathbf{W}_l \quad (10)$$

$$B_{i,l}(v_{i,1}, v_l) = \frac{b_{i,l}(v_{i,1}, v_l)}{\sum_{l=1}^L b_{i,l}(v_{i,1}, v_l)}$$

where $\mathbf{q}_{i,1}$ is the final 3D position of vertex $v_{i,1}$ for producing a new expression, $\mathbf{W}_l = [w_{l,1} \ \dots \ w_{l,M}]^T$ is the blending weight vector for control point v_l , $l = 1, \dots, L$, got with Eq. (6), and $\mathbf{W}_{i,1}$ is the final blending weights for $v_{i,1}$. We normalize all the coherency coefficients $b_{i,l}(v_{i,1}, v_l)$ of $v_{i,1}$ with respect to each control point as $B_{i,l}(v_{i,1}, v_l)$, to ensure the sum of $B_{i,l}(v_{i,1}, v_l)$ is 1.

From Eqs. (6), (8) and (10), it can be seen that the new position of each vertex of the face model after editing is represented as the linear combination of the input face meshes, and the estimation of the blending weights not only satisfies the user-inputted constraints, but also fully utilizes the natural face examples in input sequence and their spatio-temporal correlations to ensure the generation of a natural expression. Therefore, our approach can create expressions that are natural and realistic as well as achieve the user-specified goal.

4 The extension of our approach

In practical application, we will deal with a variety of different face models to create new expressions, but the pre-recorded face sets cannot cover all face models, so some models may not have the priori in the existing face sets. To further improve the practical usability of our approach, we extend our approach by combining expression cloning technique. The extended approach can effectively animate the face model whose priori is not available without expanding the existing face database. Specifically, for the face model that does not find its priori in the sets, we do not need to pre-create a full set of expressions for it as a priori; we can just utilize the existing priori embedded in the face sets to

interactively generate new expressions via combination of our approach and the expression cloning method in [22].

The basic idea of the approach extension is that (see Fig. 3), we map the user-constraints specified on the face model (we call it the target model) whose priori is not existed onto a known face model (we call it the template model) whose priori is pre-recorded in the face set; then, we perform our approach on the template model to produce a new expression based on the user-constraints and the priori embedded in the existed face set, and finally, we transfer the new expression from the template model to the target model using the expression cloning method, so as to get the final new expression on the target model.

We explain some implementation details of our extended approach. (1) We first establish dense surface point correspondences between the template model and the target model. We use the heuristic rules and RBF morphing in [22] to automatically compute dense correspondences. (2) Because the vertex displacements lead to a new expression on the model, we represent the new expression as a set of motion vector of each vertex, and expression cloning is transferring the motion vectors of the template model to each vertex of the target model. The facial geometry and proportions can vary greatly between different models, so we adjust the direction and magnitude of each motion vector when transferring them. As described in [22], the direction of the template motion vector is rotated to maintain its angle with the local surface when mapped on the target model; the magnitude of the template motion vector is scaled by the local size variations to fit to the proportion of the target model. (3) Similarly, we express the user-constraints specified on the target model as a set of motion vector of each control point (the motion vector is from the 3D position of control point to its 3D target position edited by user), and also adjust the direction and magnitude of each target motion vector when mapped onto the template model. Specifically, as shown in Fig. 3, s_l (denoted by red) is a source pixel specified by user on 2D screen and p_l (denoted by blue) is its target position. v_l' is the 3D control point selected on the target model via ray tracing. Supposing q_l' is the new position where v_l' will move to for creating a new expression, and q_l' should project onto the 2D target pixel p_l . When mapping the constraints on the target control point v_l' onto the template model, it is easy to get the corresponding template vertex v_l (i.e., the control point on the template model) with respect to v_l' based on dense surface point correspondences. Supposing q_l is the new position of v_l for generating a new expression, which is computed by estimating the blending weight of each example in the face set with Eq. (5) and linearly interpolating these examples with Eq. (3), then when transferring the motion vectors from the template model to the target model, the motion vector of v_l (denoted as $\overrightarrow{v_l q_l} = q_l - v_l$ by yellow arrow) after rotating R_l and scaling S_l , should coincide with the motion vector of

Fig. 3 The illustration of our extended approach

v_l' (denoted as $\overrightarrow{v_l' q_l'} = q_l' - v_l'$ by orange arrow). Therefore, we modify Eq. (6) as follows

$$E(W) = \theta_1 \left\| F \left(S_l \cdot R_l \cdot \left(\sum_{m=1}^M w_m v_{l,m} - v_l \right) + v_l' \right) - p_l \right\|^2 + \theta_2 \sum_{m=1}^M \left(1 + \left\| F \left(S_l \cdot R_l \cdot (v_{l,m} - v_l) + v_l' \right) - p_l \right\|^2 \right) w_m^2$$

where S_l is the rotation matrix of the template vertex v_l for adjusting the direction of the motion vector $\overrightarrow{v_l q_l}$, and R_l is the scale matrix for adjusting the magnitude of $\overrightarrow{v_l q_l}$. The expressions of S_l and R_l are described in [22] in details. Using this equation, we can transfer the user constraints from the target model to the template model and compute the optimized blending weights of examples for each template control point. Then, we use Eqs. (7), (8) and (10) to compute the new position of each template vertex to produce a new expression on the template model. Finally, we transfer the motion vectors of the template vertices to the target vertices to create a new expression on the target model. The implementation of transfer is described in [22].

The extended approach allows the user to interactively edit various face models, whether or not their priori examples are available. Therefore, our approach after extension can overcome the limitation of the face database size.

Fig. 4 Comparison with the traditional methods using PCA. **a** The user-specified control points. **b** The result by traditional methods. **c** The pre-segmented mouth region (*orange*) in traditional methods. **d** The result by our approach. **e** Three soft regions adaptively computed in our approach (respectively denoted by *green*, *orange*, and *purple*). **f** The

control points on another model. **g** The result by traditional methods. **h** The pre-segmented mouth region (*orange*) in traditional methods. **i** The result by our approach. **j** Two soft regions obtained by our approach (respectively denoted by *orange* and *purple*)

Fig. 5 Comparison of our approach and PCA method. **a, d** Are the user-specified constraints on different face models. **b, e** Are the unnatural results produced by the PCA method. **c, f** Are the natural results created by our approach

5 Experimental results

We have tested our approach with many different face data sets and compare against some other methods. We show some of the experimental results in this section.

Figure 4 shows the advantage of using the influence maps to adaptively segment the face model based on user-specified control points in our approach, compared with pre-segmenting the model into separate regions and model each region with PCA in many traditional methods [10, 11]. In Fig. 4a, three control points (red) are used to create an asymmetric smile, the control points on the left and the right corner of the mouth are respectively dragged to asymmetric target positions (blue), and the control point on the lower lip keeps its original position. Here, the face set we used contains only symmetric expressions. Figure 4b shows that when the control points are asymmetric and the expression examples are all symmetric, the traditional methods perform poorly by applying PCA on the pre-segmented mouth region (denoted by orange in Fig. 4c) to compute the maximum likelihood shape. But our approach can create good result (Fig. 4d) because it adaptively segments the mouth into three soft regions based on the user's editing, each containing a control point, and blends the soft regions based on local influence maps. Figure 4e shows the three soft regions (respectively indicated by green, orange, and purple). The influence map on each control point is denoted by color variation. The adaptive segmentation in runtime effectively decouples different parts of the mouth, and the region blending propagates the influence of each control point to the entire mesh. Figure 4f–j shows a comparison of creating an asymmetric expression

on another face model whose priori are also all symmetric expressions.

Additionally, compared to the traditional methods using PCA, our approach can create a natural expression in the case where the user's constraints are inappropriate. In Fig. 5a, with an inappropriate constraints that a control point on the lower lip is dragged to the upper lip, the traditional methods (Fig. 5b) exactly satisfy the constraints without considering whether the created expression will be reasonable, so produce an unnatural expression; our approach (Fig. 5c) although has an error to the constraints, it generates a natural expression. Figure 5d shows the constraints on another model. The traditional PCA methods have a zero error with respect to the constraints but create an unnatural and weird expression which a real person cannot make (Fig. 5e); our approach has a large error but produce a natural expression (Fig. 5f).

We also compared our approach with Zhang's method [17]. Zhang's method used normalized radial basis functions that only involve the spatial distance to compute influence map, but our approach introduces the spatio-temporal motion consistency of vertices over the face sequence as well as spatial proximity to estimate the influence map. Figure 6 demonstrates that our approach can produce more natural and realistic expressions in many cases compared to Zhang's method. With the same user-specified constraints in Fig. 6a, the result created by our approach (Fig. 6c) is more natural on the boundary of the lower lip than that created by Zhang's method (Fig. 6b). Similarly, under another user constraints in Fig. 6d, the boundary of the lower lip in Fig. 6f produced by our approach is more natural than that in Fig. 6e produced

Fig. 6 Comparison results of our approach and Zhang's method. **a, d, g, j** Are four different groups of constraints specified by the user. **b, e, h, k** Are the results produced by Zhang's method. **c, f, i, l** Are

the results produced by our approach. Some of improvements of our approach compared to Zhang's method are shown in *rectangles*

Fig. 7 The editing examples of creating a complex expression by adding control points. **a–f** Show an editing sequence of creating an expression. **g–l** Show an editing sequence for another expression

Fig. 8 The examples of editing a complex expression from a neutral face. **a–e** Shows the editing sequence of creating an symmetric expression, **f–i** shows the editing sequence of creating an asymmetric expression

Fig. 9 The expression examples created by inputting different constraints on a neutral face model. In **a–d**, the *left one* shows the face model and the user-constraints, and the *right one* shows the generated expression

by Zhang's method. Comparison between Fig. 6h created by Zhang's method and Fig. 6i created by our approach shows that our approach can satisfy the user constraints (Fig. 6g) more accurately, and produce more natural mouth shape and more fine expressional details such as folds around the mouth. Comparison between Fig. 6k, l also shows the ability of our approach to precisely match the user constraints (Fig. 6j) and create natural fine expressional details.

Figure 7 shows two editing sequences that create a complex expression from a neutral face model using our approach. The pre-recorded face set in this experiment is a spacetime mesh sequence which is reconstructed from a human's time-varying cloud points obtained at 20 frames-per-second by motion capture system. To reduce the redundancy, 192 frames are selected from all the face data to construct the pre-recorded set. The examples in the set record

various expressions of a real person; they are high-resolution and cover the expressions space in certain extent for creating complex expressions. From Fig. 7a–f, we successively add control points on the lower lip and eyebrow (the source position denoted by red and the target position by blue) to create a complex expression. Figure 7g–l shows the editing process of another complex expression.

Figure 8 shows another two editing sequences that lead from a neutral face model to a complex expression. The pre-recorded face set in this experiment comes from a sequence of face meshes edited by skilled animators that contains only symmetric expressions. The set contains 81 frames, each with natural expression, and covers as many key expressions as possible. Figure 8a–e shows an editing sequence of producing a complex symmetric expression. Figure 8f–i shows that our approach can create a complex asymmetric expression

Table 1 The geometric information and runtime of examples in Figs. 7, 8 and 9

	Frames	Vertices	Control points	Time (s) for each part of our approach			Total time (s) (on-line)
				Part 1 (on-line)	Part 2 (off-line)	Part 3 (on-line)	
Figure 7	192	23,728			13,920.66		
(a) → (b)			1	0.031		0.052	0.083
(c) → (d)			2	0.041		0.052	0.093
(j) → (k)			4	0.048		0.052	0.100
Figure 8	81	6292			829.14		
(a) → (b)			2	0.021		0.010	0.031
(b) → (c)			3	0.022		0.010	0.032
Figure 9	50	7826			1158.24		
(a)			2	0.011		0.012	0.023
(b)			3	0.011		0.012	0.023
(d)			4	0.012		0.013	0.025

Table 2 The runtime of examples (each part runs on-line) in Figs. 8 and 9

	Control points	Time (s) for each part of our approach			Total time (s) (on-line)
		Part 1 (on-line)	Part 2 (on-line)	Part 3 (on-line)	
Figure 8					
(a) → (b)	2	0.021	0.281	0.010	0.312
(b) → (c)	3	0.022	0.411	0.010	0.443
Figure 9					
(a)	2	0.011	0.351	0.012	0.374
(b)	3	0.011	0.456	0.012	0.479
(d)	4	0.012	0.562	0.013	0.587

when the input expressions are all symmetric. Figure 9 shows expressions created by specifying different constraints on the neutral face model. The pre-recorded face set in this experiment is also a sequence of face meshes edited by skilled animators. It contains 50 frames, each with a key expression for generating complex expressions. Figures 7, 8 and 9 demonstrate that our approach can produce a variety of natural expressions.

Evaluation of runtime We analyze the runtime of our approach. The program implementation of our approach consists of three parts: (1) optimizing blending weights for each control point, i.e., minimizing Eq. (6); (2) computing coherency coefficients using Eqs. (7) and (8); (3) computing the new position of each vertex using Eq. (10). In part (1), the number of control point is generally very small, and the minimization is performed with L-BFGS-B, a method that can run in real-time. In part (2), the coherency coefficients can be computed off-line. Specifically, for a given face set, we can pre-calculate the coherency coefficient between arbitrary two vertices of the face model and store them in a matrix array. The time complexity is $O(N^2M)$, where M is the number of frames in the face set, and N is the number of vertices of each frame. When the user specifies the control points in runtime,

for each control point, we can get the coherency coefficient of each vertex with respect to it by accessing the pre-stored matrix array because a control point is also a vertex. The access process is real-time. In part (3), the computation complexity is linear. Therefore, our approach can run in real-time.

We also test the runtime of our approach in our experiments. Table 1 gives the geometric information of face models in Figs. 7, 8 and 9; the runtime of each part of our approach and the total runtime when the user specified different numbers of control points in some examples of Figs. 7, 8 and 9. Table 1 shows that the time for computing coherency coefficients between arbitrary two vertices of the face model (part 2 of our approach) is large, and it depends on the number of frames and the number of vertices of frame. But the coherency coefficients can be calculated off-line and pre-stored in matrix arrays. When the algorithm runs on-line, the time of accessing the matrix arrays to get the coherency coefficients between vertex and control point is so small that it can be neglected, thus the total runtime is the sum of part 1 and part 3. It can be seen that the total time of each example in Table 1 is less than 0.1 s, demonstrating that our approach can run in real-time.

Fig. 10 The examples of animating a face model that does not have the priori in the pre-recorded face sets. In **a–d**, the *top-left one* shows the target face model whose priori is not available and the user-constraints specified on it; the *bottom-left one* shows the template face model whose priori is in the pre-recorded sets and the user-constraints transferred

from the target model to it; the *bottom-right* shows the new expression produced on the template model with the transferred user-constraints; the *top-right one* shows the new expression of the target model created by cloning the expression of the template model to it

Fig. 11 The examples of editing a face model that has no priori face data. In **a–d**, the *first one* shows the face model, and the others show the generated expressions

Masses of experiments we conducted show that, in the case that the number of vertices of each frame in a face set is less than about 9000, if the user specified fewer control points (about less than 20), the on-line runtime of part 2 that just computes the coherency coefficients between all vertices and each control point is very small. Here, the runtime of part 2 mainly depends on the number of control points and is scarcely influenced by the number of frames. Therefore, for such face sets, we do not need to pre-compute the coherency coefficients off-line, but directly compute them on-line. Table 2 shows the runtime of our approach with each part conducting on-line in Figs. 8 and 9. These times satisfy the real-time requirement of user interaction. It demonstrates that in most cases, without pre-calculation off-line, our algorithm can achieve real-time requirement.

We also tested our extended approach for the face models that do not have the prior in the existing face sets. Figure 10 shows some editing examples where the pre-recorded face set in Fig. 8 is used as a prior dataset. Figure 11 shows editing results on another face model that have no prior face set. Here, the set in Fig. 7 is used as a prior dataset. Figure 11a, b creates asymmetric expressions. Figure 11c, d creates

complex expressions with a large change. It can be seen from Figs. 10 and 11 that the expressions created on the face model with our extended approach are natural and well satisfy the user-specified goal.

6 Conclusion

We proposed an approach for generating facial expressions from the user-constraints and the pre-recorded face examples. Adaptive segmentation and blending regions based on the coherency of vertices can ensure generation of nature expressions. Our approach is simple and intuitive, and run in real-time. Additionally, we extend our approach to deal with those face models whose priori are not existed in the pre-recorded face sets. The extension allows the user to easily animate a variety of face models without expanding the existing face database, which further improves the practical usability of our approach. Our approach can find application in many fields such as natural expression synthesis in films, games, or virtual environments, and rapid face modeling for discussion or educational purposes.

Acknowledgements The work is supported by National Nature Science Foundation of China under Grant 61332015, U1609218, 61303088, 61402261, Sci-tech Development Project of Jinan City under Grant 201303021. This work is supported by The Fostering Project of Dominant Discipline and Talent Team of Shandong Province Higher Education Institutions.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Parke, F.I.: Computer generated animation of faces. In: ACM Annual Conference, vol. 1, pp. 451–457 (1972)
2. Sloan, P.-P., Rose, C., Cohen, M.F.: Shape by example. In: ACM Symposium on Interactive 3D Graphics, pp. 135–143 (2001)
3. Lewis, J.P., Cordner, M., Fong, N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In: ACM SIGGRAPH, pp. 165–172 (2000)
4. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3D faces. In: ACM SIGGRAPH, pp. 187–194 (1999)
5. Chai, J., Xiao, J., Hodgins, J.: Vision-based control of 3D facial animation. In: SCA'03, pp. 193–206 (2003)
6. Lau, M., Chai, J., Xu, Y.Q., Shum, H.Y.: Face poser: interactive modeling of 3D facial expressions using facial priors. ACM Trans. Graph. **29**(1), 89–97 (2009)
7. Lewis, J.P., Anjyo, K.-I.: Direct-manipulation blendshapes. IEEE Comput. Graph. Appl. **30**(4), 42–50 (2010)
8. Seo, J., Irving, G., Lewis, J.P., Noh, J.: Compression and direct manipulation of complex blendshape models. ACM Trans. Graph. **30**(6), 164:1–164:10 (2011)
9. Cetinaslan, O., Orvalho, V., Lewis, J.: Sketch-based controllers for blendshape facial animation. In: EG (2015)
10. Joshi, P., Tien, W., Desbrun, M., Pighin, F.: Learning controls for blend shape based realistic facial animation. In: SCA'03, pp. 35–42 (2003)
11. Zhang, Q., Liu, Z., Guo, B., Shum, H.: Geometry-driven photorealistic facial expression synthesis. In: SCA'03, pp. 177–186 (2003)
12. Li, Q., Deng, Z.: Orthogonal blendshape based editing system for facial motion capture data. IEEE Comput. Graph. Appl. **28**(6), 76–82 (2008)
13. Acquaah, K., Agada, R., Yan, J.: Example-based facial animation for blend shape interpolation. In: IEEE International Conference on Electrical, Computer and Communication Technologies, pp. 1–7 (2015)
14. Neumann, T., Varanasi, K., Wenger, S., Wacker, M., Magnor, M., Theobalt, C.: Sparse localized deformation components. ACM Trans. Graph. **32**(6), 2504–2507 (2013)
15. Ma, X., Le, B.H., Deng, Z.: Style learning and transferring for facial animation editing. In: SCA'09, pp. 123–132 (2009)
16. Tena, J.R., Torre, F.D., Matthews, I.: Interactive region-based linear 3D face models. In: ACM SIGGRAPH, Article 76 (2011)
17. Zhang, L., Snavely, N., Curless, B., Seitz, S.M.: Spacetime faces: high resolution capture for modeling and animation. ACM Trans. Graph. **23**(3), 548–558 (2004)
18. Akhter, I., Simon, T., Khan, S., Matthews, I., Sheikh, Y.: Bilinear spatiotemporal basis models. ACM Trans. Graph. **31**(2), Article 17 (2012)
19. Seol, Y., Lewis, J.P., Seo, J., Choi, B., Anjyo, K., Noh, J.: Spacetime expression cloning for blendshapes. ACM Trans. Graph. **31**(2), Article 14 (2012)
20. Seol, Y., Seo, J., Kim, P.H., Lewis, J.P., Noh, J.: Weighted pose space editing for facial animation. Vis. Comput. **28**(3), 319–327 (2012)
21. Xu, F., Chai, J., Liu, Y., Tong, X.: Controllable high-fidelity facial performance transfer. ACM Trans. Graph. **33**(4), 1–11 (2014)
22. Noh, J., Neumann, U.: Expression cloning. In: ACM SIGGRAPH, pp. 277–288 (2001)

Jing Chi is currently an Associate Professor of Department of Computer Science and Technology, Shandong University of Finance and Economics. She received her Ph.D. in Computer Science from Shandong University of China in 2012. Her research interests include computer facial animation, dynamic modeling, and curve fitting.

Shanshan Gao received her Ph.D. from Shandong University, China in 2011. She is currently an Associate Professor of department of Computer Science and Technology, Shandong University of Finance and Economics. Her research interests include computer graphics, image saliency detection and image segmentation.

Caiming Zhang is a Professor and doctoral supervisor of School of Computer Science and Technology at Shandong University. He received his Dr Eng degree in Computer Science from Tokyo Institute of Technology, Japan, in 1994. From 1997 to 2000, He has held a visiting position at University of Kentucky, USA. His research interests include CAGD, CG, information visualization, and medical image processing.