ORIGINAL ARTICLE

Weighted pose space editing for facial animation

Yeongho Seol · Jaewoo Seo · Paul Hyunjin Kim · J.P. Lewis · Junyong Noh

Abstract Blendshapes are the most commonly used approach to realistic facial animation in production. A blendshape model typically begins with a relatively small number of blendshape targets reflecting major muscles or expressions. However, the majority of the effort in constructing a production quality model occurs in the subsequent addition of targets needed to reproduce various subtle expressions and correct for the effects of various shapes in combination. To make this subsequent modeling process much more efficient, we present a novel editing method that removes the need for much of the iterative trial-and-error decomposition of an expression into targets. Isolated problematic frames of an animation are re-sculpted as desired and used as training for a nonparametric regression that associates these shapes with the underlying blendshape weights. Using this technique, the artist's correction to a problematic expression is automatically applied to similar expressions in an entire sequence, and indeed to all future sequences. The extent and falloff of editing is controllable and the effect is continuously propagated to all similar expressions. In addition, we present a search scheme that allows effective reuse of pre-sculpted editing examples. Our system greatly reduces time and effort required by animators to create high quality facial animations.

Keywords Pose space · Blendshape · Facial animation · Editing

Y. Seol · J. Seo · P.H. Kim · J. Noh (✉)
KAIST, Daejeon, Korea
e-mail: junyongnoh@kaist.ac.kr

J.P. Lewis
Weta Digital and Victoria University, Wellington, New Zealand

1 Introduction

Blendshape based facial animation is a commonly used technique in production as exemplified by the movies such as *The Lord of the Rings*, *King Kong*, and *Avatar*. The advantage of this approach is that artists can create an easily understandable parameterization of facial expression by directly sculpting target shapes that are added to produce the final expression. The blendshape representation is commonly used in animation retargeting, and a number of solving and retargeting methods have been introduced [5–7, 9, 16]. The goal of these methods is to find the combination of blendshape targets that best produces a given facial expression, often one obtained from motion capture. The results of automated retargeting methods is a good starting point for facial animation for feature films, however, high quality animation inevitably involves further work by animators to refine the raw retargeting results.

A blendshape model typically begins with a set of targets reflecting the major facial muscles or expressions. When animators are asked to edit existing blendshape based facial animations (such as those obtained by retargeting motion capture), they re-adjust the blendshape weights until the desired results are obtained. However, with a new model it is often the case that no combination of the existing shapes will produce the desired expression. The next stage of editing is to sculpt additional details to augment the model's expressiveness or to remove unintended interference caused by the combination of blendshapes [19]. This secondary editing stage is very important in creating a high quality facial animation. It can also involve considerable effort, since a model with M primary targets can have $\binom{M}{2} \approx O(M^2)$ pairwise interactions that need to be examined and possibly corrected, and in some cases triads or even higher order interactions must be considered. For example, the *Gollum* character from *The Lord of the Rings* films required almost 1000

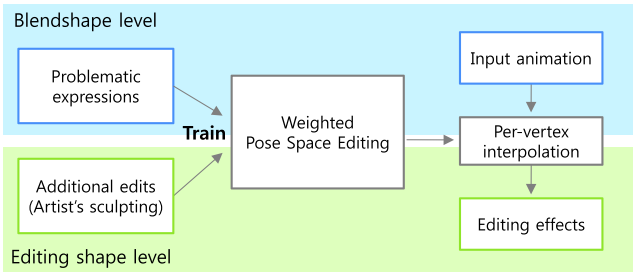


Fig. 1 The overall process of facial animation editing with our method. The weights of blendshapes drive the editing shapes using weighted pose space editing

blendshapes and 80 to 90% of the shapes were created in this stage [19]. One of the most difficult aspects of this process is refining a complex blendshape model to produce a desired expression. This involves an iterative process of choosing a candidate subset of target shapes that will generate the new expression, re-sculpting these targets, and verifying that the changes do not overly interfere with other expressions.

We present a novel facial animation editing method for blendshape models that specifically targets this shape refinement phase, largely eliminating the expensive iterative re-sculpting needed to achieve these secondary effects. We consider the following to be the desirable characteristics of a new editing system:

(a) Once the artist edits a problematic expression, similar expressions should be edited automatically.
(b) The effect of editing should be propagated smoothly through the surrounding frames with similar expressions to maintain the temporal coherency of the original sequence.
(c) The method should naturally integrated into the animation pipeline and be intuitive for artists.

We formulated the novel editing method as a non-parametric regression problem. As a set of blendshape weights is the main control for facial animation, it is useful to define a mapping between the blendshape weights and additional sculpted edits. The influence of each blendshape on each face vertex is analyzed prior to the training of a Radial Basis Function (RBF) network with the Gaussian kernel for each vertex. Figure 1 illustrates the overall process of the facial animation editing method. Given training examples (i.e. problematic expressions and additional edits), the RBF network (i.e. weighted pose space editing) is constructed, and it automatically produces editing effects for similar expressions within an input animation.

We applied this technique to existing facial animation sequences produced either by manual keyframing or by animation retargeting. Our technique efficiently accomplished the desired editing and the animators found it very intuitive. Our GPU based implementation allowed interactive performance even for a production character with a large number of vertices/blendshapes. In addition, we introduce a search method that enables effective reuse of pre-sculpted edits for a new facial animation sequence. Finally, we present a thorough comparison between our method and the predominant current approach, the so-called "Fix Shape Method" (FSM), to show that our method performs much more efficiently and greatly reduces the required manual work.

2 Related work

Previous blendshape based facial animation editing approaches have mainly focused on editing data-driven facial animation [7, 9, 16] or providing intuitive control [3, 12, 14] to create desired expressions. Choe et al. [5] and Li and Deng [17] use spline curves to smoothly interpolate additional user edits on top of the original animation. Lewis et al. [15] describe an editing technique that provides better control of interference effects within an existing blendshape model. Intuitive direct manipulation of linear facial models is described in [12, 14]. High-level editing methods for speech animation are introduced in [3, 6]. However, all the methods mentioned above concentrate on adjusting existing blendshapes' weights to produce desired facial expressions, which is not the main focus of this paper as explained in Sect. 1.

Ma et al. [18] use a constraint-based Gaussian process model to reduce repetitive editings on similar expressions. Our new editing technique is similar in concept with this work. Both methods depend on a machine learning technique to learn the artist's editing style. However, their method targets an automatically generated linear ICA parameterization of the face, whereas our approach targets the manually sculpted but nonlinear FSM blendshape scheme used in industry practice.

Commercial animation packages include a blendshape engine as a standard component. Artists are familiar with augmenting extra details and correcting interference by using the FSM [19]. In this approach, additional sculpted blendshape targets termed "fix shapes" are activated by the combination of several blendshapes. As the FSM has been the standard production solution for our chosen problem, we provide a rigorous comparison between our method and FSM in Sect. 5.1.

A pose space approach has been widely employed for character deformation and facial animation (e.g. [10, 13]). Kurihara and Miyata [11] introduce weighted pose space deformation formulated using normalized RBFs in order to construct a realistic deformable hand model. Bickel et al. [1] employ a weighted pose space approach to augment extra wrinkle and pore details on the face. Our work and Bickel et al. [1] share similar weighted pose space formulations, but their focus is on marker based data-driven facial animation.

3 Method

3.1 Blendshape model

A brief description of blendshape model is presented. A blendshape based face model with V vertices and M blendshapes is notated as follows:

$$\mathbf{f} = \mathbf{f}_0 + \mathbf{B}\mathbf{w} \tag{1}$$

where $\mathbf{f}$ is the face model represented as a column vector $\in \mathbb{R}^{3V}$ consisting of position values of vertices in a consistent order xyzxyzxyz... and $\mathbf{f}_0$ is the same size vector for the neutral face. $\mathbf{B}$ is a $3V \times M$ blendshape matrix whose columns represent deltas of each blendshape from the neutral position. $\mathbf{w}$ is a column vector $\in \mathbb{R}^M$ of blendshape weights. We choose the delta blendshape representation [15] over the whole-face representation for this paper, however, an equivalent formulation for the whole-face blendshape representation is straightforward.

3.2 Pose space editing

Additional vertex level editing after the adjustment of blendshape weights is necessary for the creation of high quality animation. For the convenience of animators (and in common with the FSM approach), it would be a good strategy to let the M blendshape weights that animators use for manual keyframing drive the additional editings. The desired functionality can be obtained by a formulation using non-parametric regression. The blendshape weights are the domain and per-vertex displacements from the initial positions in local frames are the range of this interpolation.

In our Pose Space Editing (PSE) algorithm, RBF with the Gaussian kernel is chosen as the regression method. The editing function $\mathbf{d} : \mathbb{R}^M \to \mathbb{R}^{3V}$ has the form

$$\mathbf{d}(\mathbf{w}) = \sum_{i=1}^{P} \mathbf{x}_i \phi\big(\|\mathbf{w} - \mathbf{w}_i\|\big) \tag{2}$$

$$\phi\big(\|\mathbf{w} - \mathbf{w}_i\|\big) = \exp\left(\frac{-\|\mathbf{w} - \mathbf{w}_i\|^2}{2\sigma^2}\right)$$

where P is the number of manually edited frames, $\mathbf{w}_i$ is the blendshape weight vector of ith edited pose, $\mathbf{x}_i$ is a $\mathbb{R}^{3V}$ vector of RBF weights, and σ is a user parameter which controls the range of editing influence around the example poses. Given the edited poses and vertex displacements, (2) can be rewritten as

$$\mathbf{D} = \mathbf{X}\mathbf{\Phi} \tag{3}$$

where $\mathbf{D}$ is a $3V \times P$ matrix whose columns represent vertex displacements of each example edit, $\mathbf{X}$ is a $3V \times P$ matrix with $\mathbf{x}_i$ in its columns, and $\mathbf{\Phi}$ is a $P \times P$ matrix with $\mathbf{\Phi}_{i,j} = \phi(\|\mathbf{w}_i - \mathbf{w}_j\|)$. The RBF weights can be obtained by multiplying the inverse of $\mathbf{\Phi}$ on both sides. The parameter σ can be optimized as explained in the neural net literature [2], but we instead provide it as a user parameter. In many cases, animators want control over range of influence of each edit, and the animatable σ serves the role very well. Figure 2 exhibits the different effects possible by controlling the value of the range parameter σ.

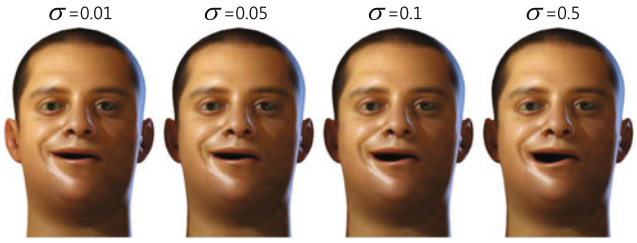


Fig. 2 Facial expressions with the different σ values

3.3 Weighted pose space editing

In this basic form of PSE, there is a combinatorial explosion of the number of needed edits. For example, the weights in a pose with the eyes open are distant from the weights reflecting a pose with the eyes closed. This is as expected, but it means that an edit of the mouth area with the eyes open will disappear as the eyes are closed. As a result, the required editing examples increase drastically to sufficiently cover all variations of every local region. As a solution to this problem, we implemented localized Weighted Pose Space Editing (WPSE) that works in a per-vertex manner as in [1, 11].

Our localized WPSE algorithm is obtained by redefining the Euclidean metric $\|\mathbf{w} - \mathbf{w}_i\|$ in (2) in a per-vertex manner

$$\|\mathbf{w} - \mathbf{w}_i\|_v := \left(\sum_{j=1}^{M} \alpha_{v,j}(w_j - w_{j,i})^2\right)^{1/2} \tag{4}$$

where $w_{j,i}$ is the jth blendshape weight of ith modified pose. The value of α is assigned based on the impact of a blendshape on that vertex. Specifically,

$$\alpha_{v,j} = \frac{l_{v,j}}{\sum_{j=1}^{M} l_{v,j}} \tag{5}$$

where $l_{v,j}$ is a per-vertex Euclidean distance from the neutral expression to each blendshape. When $\alpha_{v,j}$ is near zero, we intentionally add a very small ε to avoid a singular basis matrix.

The RBF is also reformulated for each vertex to take WPSE into account. The new equation is slightly different from (2):

$$\mathbf{d}_v(\mathbf{w}) := \sum_{i=1}^{P} \mathbf{x}_{v,i} \phi\big(\|\mathbf{w} - \mathbf{w}_i\|_v\big) \tag{6}$$

During the training phase, the RBF weight computation involves solving $P \times P$ systems for every edited vertex. The result is $3P$ weights to be stored per vertex. Although this is a considerable increase in the size of the model, it is justified for high quality work and easily calculated within the memory of a modern GPU. The $\alpha_{v,j}$ in (5) are also computed during this phase and employed throughout the rest of the editing process. At run-time, (6) is evaluated at every vertex to generate a final result, given a new facial expression. The computation can be easily parallelized over all vertices, allowing a highly efficient GPU implementation.

4 Searching and reusing pre-sculpted edits

For a character that is repeatedly used in hundreds of sequences, it is highly likely that desired edits for a new sequence were already sculpted during the handling of previous sequences. When a problematic expression is identified in the new sequence, the artist has to choose between sculpting a new example shape and reusing previous edits. Although reuse is desirable in general, the permanent addition of sequence-specific edits to the model imposes unnecessarily heavy computational load and increases the risk of overfitting (which will be discussed in Sect. 5.2). Employing only a compact set of necessary edits for WPSE makes the editing stable and efficient.

We present a simple but effective search method that can be applied in the space of hundreds of existing edits. The method searches through previous editing examples that were created for poses similar to the current problematic expression. The search results guide an artist to decide whether to use one of the recommended edits or to sculpt a new one.

The artist can specify a region of interest (ROI) to indicate the area that requires a new edit. Among the existing shapes, the shapes recommended for reuse are those with a value E less than a threshold τ in the ROI,

$$E = \sum_{v=1}^{V'} \left(\sum_{j=1}^{M} \alpha_{v,j} \left(w_j - w_j^p \right)^2 \right)^{1/2} \tag{7}$$

where V' is the number of vertices in ROI and $\alpha_{v,j}$ are the same weight values as the ones computed in (5). w_j represents the jth blendshape weight on the current face model and w_j^p represents the same weight on the pth existing edit. τ is a user parameter, subject to $\tau \in [0, \max E]$, which allows the animator to check recommended existing edits by varying the value. Figure 3 shows the user specified ROI on a pose and recommended edits selected from the database of 452 existing shapes for different τ values.

The artist examines the result obtained by including the recommended edits and retains them in the current WPSE session if the result is satisfactory. When no existing edits provide a desired editing result, a new shape is sculpted. The new edit is used for the current sequence and also stored in the database for future use. Augmented by this search method, WPSE can be effectively employed for the fast creation of high quality animation while maintaining interactive updates and animation preview.

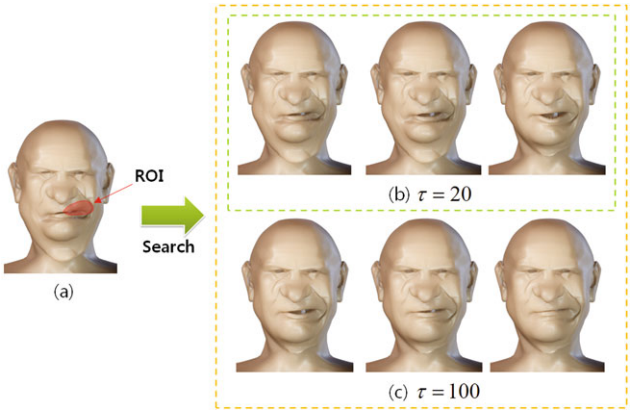


Fig. 3 Search results corresponding to different τ values (**b**, **c**). **a** represents the target expression with a user specified ROI

5 Results and discussion

Our editing method facilitates the creation of high quality animation. Additional editing of existing blendshape based facial animation becomes easy as the only task to be done is sculpting desired edits at problematic frames and pressing the *Update* button. There is no need to decompose a desired expression into a set of candidate blendshape targets and carefully re-sculpt these so they sum to reproduce the desired expression. The animatable user control parameter σ that determines the interpolation range of WPSE proves useful. Overall, the method is very well received by professional animators because it is well adapted to the traditional facial animation process while removing the most tedious steps from it.

Figure 4 shows the GUI of our implementation. Using the GUI, artists can conveniently add or remove editing examples. Two ways of adding a new editing example are available: (1) direct sculpting on a problematic frame and (2) training pre-sculpted editing examples. Furthermore, the range of editing effects is easily controlled by the animatable σ. Modifying σ has a non-linear effect as shown in Fig. 2, while animators prefer a controller that makes visually linear deformation effect. To provide animators with an intuitive control, we devised a mapping between σ from (2) and σ_G on the GUI. In (6), it is seen that the displacement of a vertex $\mathbf{d}_v$ and σ are related as $\mathbf{d}_v \propto \exp(\frac{-1}{\sigma^2})$. The value of σ is determined as follows: $\sigma = \lambda\sqrt{-1/\log \sigma_G} + c$. λ and c are constants which are employed to make $\sigma = \sigma_G$, when σ_G is 0.01 or 0.99.

Figure 5 shows editing results by WPSE for different face models. Several animations were obtained; these included both manually created keyframe animations and animations resulting from a data-driven retargeting method. Editing applied to one expression (the leftmost column) is effectively recreated on locally similar expressions in the same sequence. We intentionally applied large and noticeable edits for the purpose of visualization. In a real situation, most edits would be very subtle because the purpose of editing is to augment additional details or correct undesired interference. The face model used in the left side of Fig. 5 has 3k vertices and 35 blendshapes. A high quality production model in the right side has 40k vertices and 138 blendshapes. Figure 6 shows temporally coherent editing effect over an animation sequence of 12 frames. The editing effect produced on frame 68 was smoothly propagated across the surrounding frames. Please look at the accompanying video clip for the full animation sequences.

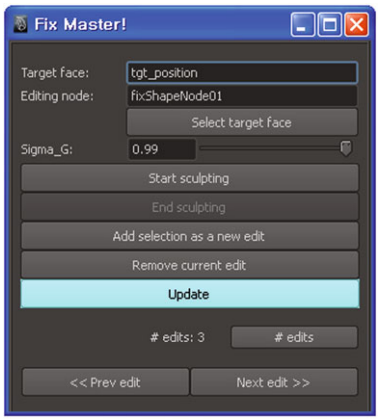


Fig. 4 The GUI for WPSE. As is the case with blendshapes, the power of this technique arises from directly sculpted examples, a capability already provided by the underlying animation software

The capability of PSE and WPSE is analyzed and reported in Fig. 7. The first column (a) presents the target expressions and corresponding additional vertex level editing. Both PSE and WPSE successfully produced proper results when a new input expression is globally similar to the target expressions as shown in the second column (b). In the third column (c), the new input expressions are locally different from the target expressions (Look at the eye region). Whereas PSE fails to generate a proper shape, WPSE still reproduces the intended editings.

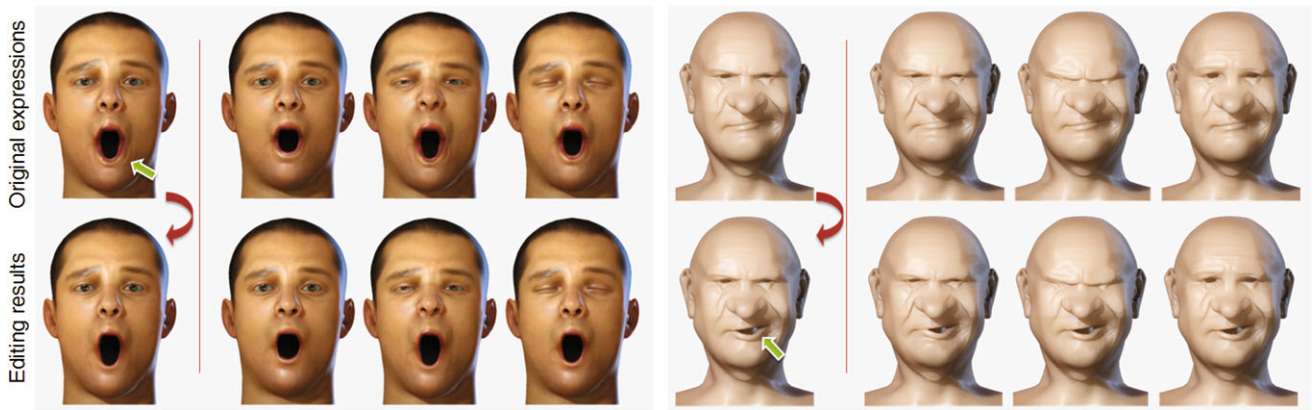


Fig. 5 The *leftmost columns* of each face model represent a target expression and the additional editing. WPSE successfully regenerates editing effect on locally similar expressions (the three pairs of faces in the right). *Green arrows* indicate the edited regions

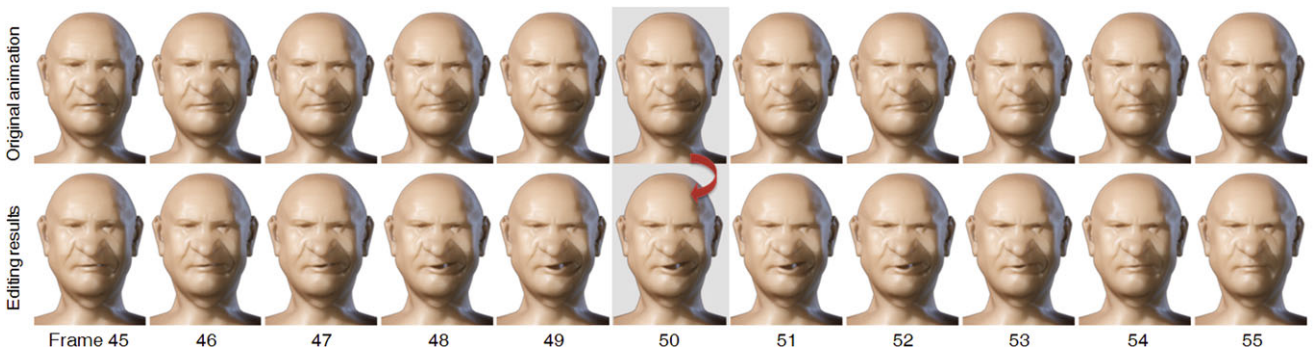


Fig. 6 The *first row* shows the original animation and the *second row* shows the results after the editing. At frame 50, the expression is edited and its influence is smoothly propagated across the surrounding frames

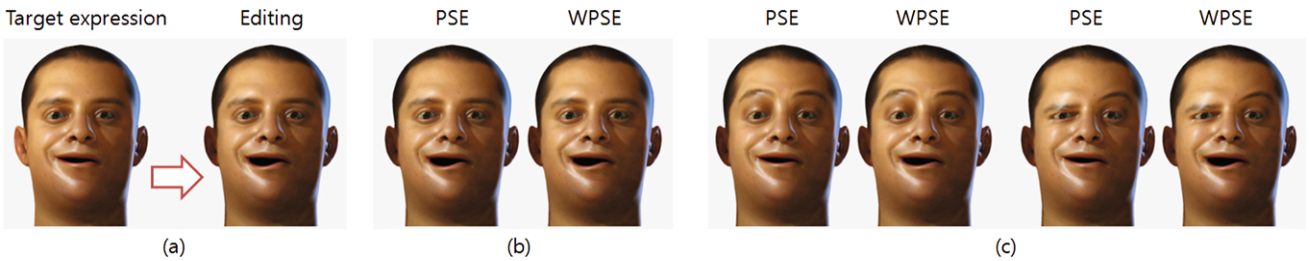


Fig. 7 Performance of PSE and WPSE is compared. **a** represents a target expression and its vertex level editing. When a new input face has a globally similar expression to the target expression (**b**), both PSE and WPSE work properly. However, if the input face has a different local expression from the target expression (**c**), PSE is no longer activated. In contrast, WPSE still generates the desired corrections

Table 1 Performance of WPSE with and without GPU implementation. "# Moved vertices" represents the number of vertices actually displaced by edits

	# Edits	# Moved vertices	CPU		GPU	
			Training phase	Run-time	Training phase	Run-time
Face 1	10	1,427	0.1 s	8.5 ms	0.06 s	0.07 ms
(3k vertices)	50	2,040	3.34 s	25.4 ms	2.27 s	0.65 ms
Face 2	10	32,343	3.06 s	0.23 s	1s	2.3 ms
(40k vertices)	50	35,740	108.57 s	1.26 s	49.6 s	14 ms

WPSE allows an efficient parallel GPU implementation. During the training phase, the basis matrix computation and matrix inversion for all edited vertices were parallelized by the GPU and CPU, respectively. At run-time, both basis matrix computation and RBF interpolation were computed by GPU. Table 1 reports the performance of WPSE tested with and without GPU parallelization. When the test model has a relatively small number of vertices ($V = 3$k), WPSE offered real-time performance with only CPU implementation. When the production model ($V = 40$k) was used, the CPU implementation failed to provide real-time interaction, however, WPSE could still achieve an interactive frame rate of 14 ms/frame with $P = 50$ training expressions by the GPU parallelization. The test was carried out on a standard PC with 8-core Intel Xeon 2.8 GHz, NVIDIA GeForce GTX 580, and 8GByte of memory. This experiment verifies that WPSE can be effectively employed in a practical situation.

5.1 Comparison with the fix shape method

The FSM is the commonly used production solution for facial animation editing. The WPSE and FSM have the same input and output: in both methods, existing blendshapes' weights drive additional vertex displacements. In the FSM, these vertex displacements are represented in the form of a new blendshape, which is a "fix shape". Figure 8 shows the example of rigging network of fix shapes commonly practiced by artists [19].

Each fix shape is triggered by two or more blendshapes. The weight value of a fix shape is obtained by multiplying the weight values of each trigger blendshape,

$$fixShape.weight = \prod_k triggerShape_k.weight \tag{8}$$

One of the difficulties of using the FSM is choosing appropriate trigger shapes out of many blendshapes, as a facial pose that requires correction is almost always the combination of many shapes. It is relatively easy to select proper trigger shapes when the model has fewer than 20 blendshapes. However, if the number of blendshape becomes hundreds, designing an appropriate rigging network of fix shapes is basically an iterative trial and error process that requires much insight and patience. Note that the number of possible trigger combinations to consider is (of course) combinatorial. In contrast, WPSE intelligently builds an adequate trigger structure, so artist do not need to iteratively search for a suitable combination of triggers.

In addition, because the FSM is based on simple multiplication, a new fix shape can be sculpted only for an expression which has only 0 or 1 as its blendshapes' weight. (i.e. fix shapes are always situated at corners of the weight hypercube). This is a big disadvantage of the FSM because problematic frames rarely have such weight combination. As a result, direct sculpting on problematic expressions is not possible and an artist has to infer appropriate trigger shapes and vertex level editing through an intensive trial and error process. Lastly, the onset of a fix shape is quadratic when two triggers are active, cubic when three triggers are used, etc. When three or more weights are active, this results in very little effect when the weights are low, followed by an overly rapid rise. In all cases, if the weights

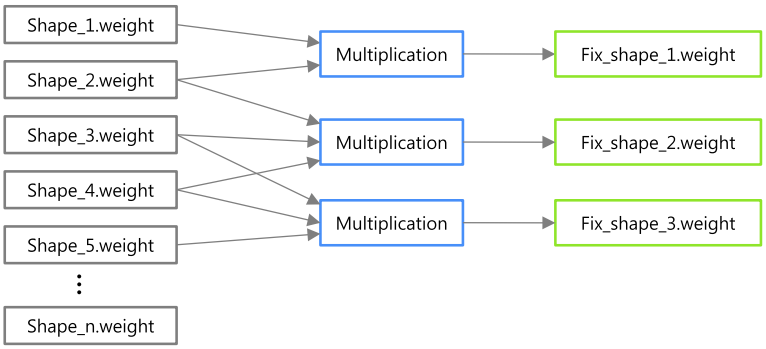


Fig. 8 The rigging network of fix shapes

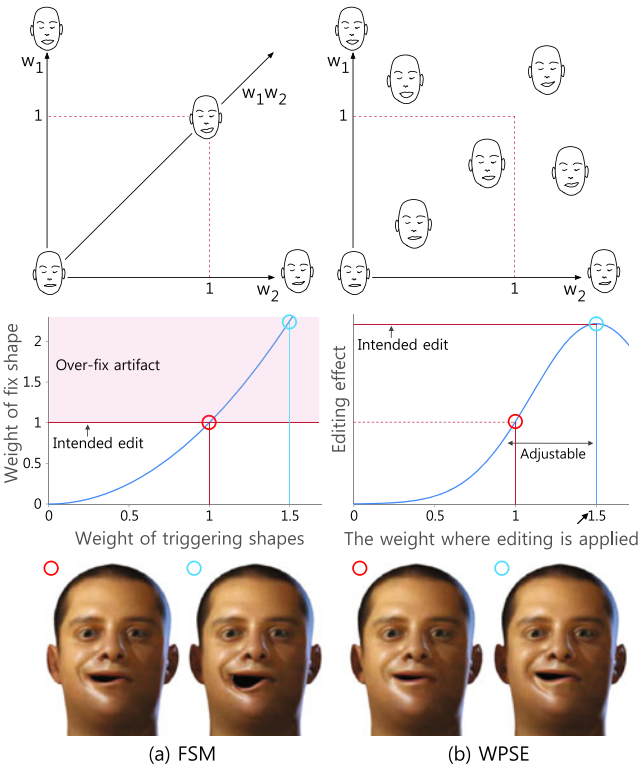


Fig. 9 Abstract comparison between the **a** FSM and **b** WPSE. (*Top-left*) In the FSM, sculpted targets *must* be placed at the value 1 on each parameter (i.e. at the vertices of a hypercube in the parameter space). The activation graph (*middle-left*) shows that the extrapolation is quadratic (in the case of a shape triggered by two parameters), which is undesirable. The WPSE technique allows shapes to be placed directly at any points in the parameter space that require correction (*top-right*), thus leading to better performance, fewer required shapes, and controllable extrapolation

greater than one are used (which frequently happens during both automatic retargeting and manual keyframing), the fix shapes "explode" (Fig. 9). In contrast, WPSE provides interactive editing capability by allowing a new edit on any desired point in the parameter space and is free from over-fix artifact. Figure 9 shows abstract comparison of activation graphs between the FSM and WPSE, and the resulting expressions by both methods. The FSM can create a new fix only on the red circle (where all weights of the trigger shapes are 1), and generates over-fix artifact on extrapolation (the blue circle). In contrast, as the weight selection is automated for WPSE, a new fix for the blue circle can be sculpted. As a result, a proper expression can also be generated at the red circle by adjusting its range parameter σ. We assumed that two trigger shapes were used and their weight changed from 0 to 1.5. WPSE shows smooth and adjustable editing effects without "explosions" in overshoot situations.

5.2 Limitations and future work

Because WPSE is derived from the RBF network, it is not completely free from the problem of overfitting [8]. If several distinct edits are sculpted on very similar poses, editing results can be inconsistent. This is illustrated in Fig. 10, where we intentionally trained two distinct editing examples with similar poses, and WPSE led to an undesirable interpolation result. To remedy the situation, we devised a partitioning scheme that partitions the WPSE system into several layers depending on each example's proximity in pose space. For instance, if a new edit is sculpted within the threshold distance η from existing edits, the new edit is placed on an additional layer (an additional independent RBF is trained). The overfitting artifact shown in Fig. 10(d) is avoided when this partitioning method is employed (Fig. 10(e)). Independent control of σ for different layers is another benefit of

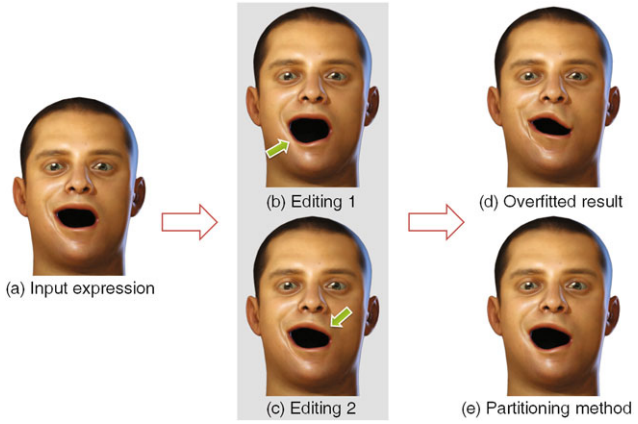


Fig. 10 When distinct edits (**b**, **c**) are trained on similar poses, WPSE may generate unintended overfitting result (**d**). In such cases, a partitioning method prevents the overfitting (**e**). The *green arrows* indicate the edited regions

this scheme. Nevertheless, it is also desirable for the artists to understand the underlying problem.

The search method in Sect. 4 can be more efficient with the utilization of an accelerating data structure for editing examples. For the PSE case, storing the pose of edits into a kd-tree would expedite the search by quickly finding the nearest neighbors. However, the same approach may not work properly in the case of WPSE, as each vertex considers the pose of edits differently depending on their α (5). By averaging the α values of all vertices in the ROI, the search result from kd-tree may approximate the actual result calculated by (7). Developing an efficient data structure that supports the WPSE is a future extension of this work.

In this work, we employed the Gaussian kernel for RBF interpolation. Although it works adequately in our experience, it would be useful to test other RBF kernel functions. A common choice are the n-harmonic splines, which minimize the integral of the squared nth derivatives [4]. This approach involves solving a system that is augmented by a polynomial basis spanning the null space of the spline operator. Note that the choice of a Gaussian kernel corresponds to minimizing a particular weighted sum of *all* orders of derivatives [20].

6 Conclusion

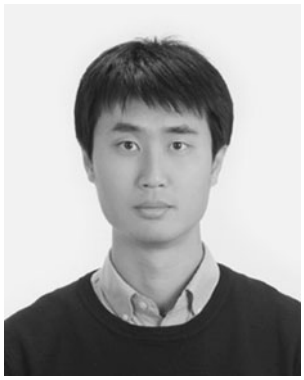
This paper presented a practical facial animation editing technique suited to blendshape model for the generation of high quality animation. Given an existing blendshape animation, problematic frames are re-sculpted by an artist and the corrections are learned with a non-parametric regression. The artist can easily add or remove edits and control the amount of modification. The search scheme for pre-sculpted edits expedites the editing process when a large number of sequences of facial animation have to be created by the repetitive use of similar edits. The results verify that WPSE outperforms the current production solutions, particularly by eliminating the trial-and-error decomposition required by the FSM. Simultaneously, WPSE shares the assumptions of current production practice, allowing easy adoption.

Acknowledgements We wish to thank Younghui Kim for shading/lighting, Jason Osipa for the face model, and the reviewers for their time and comments. This work was supported by KOCCA/MCST (2-10-7602-003-10743-01-007, Software Development for 2D to 3D Stereoscopic Image Conversion).

References

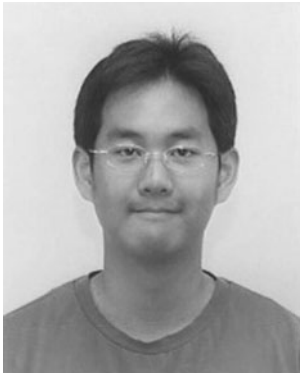
1. Bickel, B., Lang, M., Botsch, M., Otaduy, M.A., Gross, M.: Pose-space animation and transfer of facial details. In: SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 57–66. Eurographics Association, Aire-la-Ville (2008)
2. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, London (1995)
3. Cao, Y., Faloutsos, P., Pighin, F.: Unsupervised learning for speech motion editing. In: SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 225–231. Eurographics Association, Aire-la-Ville (2003)
4. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3d objects with radial basis functions. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01, pp. 67–76. ACM, New York (2001)
5. Choe, B., Lee, H., Seok Ko, H.: Performance-driven muscle-based facial animation. J. Vis. Comput. Animat. **12**, 67–79 (2001)
6. Chuang, E., Bregler, C.: Mood swings: expressive speech animation. ACM Trans. Graph. **24**(2), 331–347 (2005)
7. Deng, Z., Chiang, P.Y., Fox, P., Neumann, U.: Animating blendshape faces by cross-mapping motion capture data. In: I3D '06: Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, pp. 43–48. ACM, New York (2006)
8. Fornberg, B., Flyer, N.: The Gibbs phenomenon for radial basis functions. In: Gibbs Phenomenon in Various Representations and Applications. Sampling, Potsdam (2006)
9. Joshi, P., Tien, W.C., Desbrun, M., Pighin, F.: Learning controls for blend shape based realistic facial animation. In: SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 187–192. Eurographics Association, Aire-la-Ville (2003)
10. Komorowski, D., Melapudi, V., Mortillaro, D., Lee, G.S.: A hybrid approach to facial rigging. In: ACM SIGGRAPH ASIA 2010 Sketches, SA '10, pp. 42:1–42:2. ACM, New York (2010)
11. Kurihara, T., Miyata, N.: Modeling deformable human hands from medical images. In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '04, pp. 355–363. Eurographics Association, Aire-la-Ville (2004)
12. 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), 1–17 (2009)
13. Lee, G., Hanner, F.: Practical experiences with pose space deformation. In: ACM SIGGRAPH ASIA 2009 Sketches. ACM, New York (2009)
14. Lewis, J., Anjyo, K.: Direct manipulation blendshapes. Comput. Graph. Appl. **30**(4), 42–50 (2010). Special issue: Digital Human Faces

15. Lewis, J.P., Mooser, J., Deng, Z., Neumann, U.: Reducing blendshape interference by selected motion attenuation. In: I3D '05: Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games, pp. 25–29. ACM, New York (2005)
16. Li, H., Weise, T., Pauly, M.: Example-based facial rigging. ACM Trans. Graph. **29**(4), 1–6 (2010)
17. Li, Q., Deng, Z.: Orthogonal blendshape based editing system for facial motion capture data. IEEE Comput. Graph. Appl. 76–82 (2008)
18. Ma, X., Le, B.H., Deng, Z.: Style learning and transferring for facial animation editing. In: SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 123–132. ACM, New York (2009)
19. Osipa, J.: Stop Staring: Facial Modeling and Animation Done Right. Wiley, New York (2007)
20. Yuille, A., Grzywacz, N.: A mathematical analysis of the motion coherence theory. Int. J. Comput. Vis. **3**(2), 155–175 (1989)



Yeongho Seol is currently a Ph.D. student in the Graduate School of Culture Technology at KAIST. His research interests include facial animation, character animation, and interactive media. Besides his research, he has participated animation productions as a rigging artist. He earned his B.S. from Industrial Design, KAIST and M.S. from GSCT, KAIST. He worked as a research intern at Weta Digital.

Jaewoo Seo received his B.S. degree in Digital Media in 2005 from Ajou University, Korea, and his M.S. degree in Culture Technology in 2007 from Korea Advanced Institute of Science and Technology (KAIST). He is currently working toward his Ph.D. degree at KAIST. His research interest includes rigging, animation, and deformation of virtual characters.



Paul Hyunjin Kim received his B.S. degree in Mechanical Engineering in 2009 from the Korea University, Korea, and his M.S. degree in Culture Technology in 2011 from Korea Advanced Institute of Science and Technology (KAIST). He has currently been accepted to the Ohio State University for his Ph.D. degree. His research interest includes facial animation, retargeting, and blendshape.

J.P. Lewis is a research lead at Weta Digital and Senior Lecturer at Victoria University in New Zealand. In the past, he has worked as a Research Associate at the University of Southern California and Stanford University, and in the film industry at Industrial Light and Magic, Disney, and elsewhere. Lewis has published more than 40 papers on a variety of topics and has several patents and film credits.

Junyong Noh is an Associate Professor in the Graduate School of Culture Technology at the Korea Advanced Institute of Science and Technology (KAIST). He is also affiliated with KAIST Institute of Entertainment Engineering (KIEE). He earned his computer science Ph.D. from the University of Southern California (USC) in 2002 where his research focus was on facial modeling and animation. His research relates to human facial modeling/animation, character animation, fluid simulation, and stereoscopic visualization. Prior to his academic career, he was a graphics scientist at a Hollywood visual effects company, Rhythm and Hues Studios. He performed R&D for movie post productions including Superman Returns, Happy Feet, The Chronicles of Narnia, Garfield, Around the World in 80 Days, and The Chronicles of Riddick. He had also participated in the implementation of fluid simulation software, which received an academy award in 2008. He has been doing consulting for or collaborative work with many companies such as Weta Digital, SKT, ETRI, Macrograph Olive Studio, and KAI Studio.