



Interactive Facial Expression Editing with Non-linear Blendshape Interpolation

Ji Hyun Roh^{†1} Seong Uk Kim^{†2}  Hanyoung Jang³ Yeongho Seol⁴ Jongmin Kim^{‡1,2} 

¹Department of Computer Science and Engineering, Kangwon National University, South Korea

²Interdisciplinary Graduate Program in Medical Bigdata Convergence, Kangwon National University, South Korea

³Game AI Lab, NC, South Korea

⁴NVIDIA, South Korea

Abstract

The ability to manipulate facial animations interactively is vital for enhancing the productivity and quality of character animation. In this paper, we present a novel interactive facial animation editing system that can express the naturalness of non-linear facial movements in real-time. The proposed system is based on a fully automatic algorithm that maintains all positional constraints while deforming the facial mesh as realistic as possible. Our method is based on direct manipulation with non-linear blendshape interpolation. We formulate the facial animation editing as a two-step quadratic minimization and solve it efficiently. From our results, the proposed method produces the desired and realistic facial animation better compared to existing mesh deformation methods, which are mainly based on linear combination and optimization.

CCS Concepts

• **Computing methodologies** → Facial Animation; Performance Capture; Blendshape; Mesh Deformation;

1. Introduction

An interactive interface tool for facial animation editing is crucial for the creation of realistic and natural characters. Several different types of existing commercial editing tools and related applications have been introduced and widely used in film and game production due to the growing popularity and attraction of utilizing virtual reality technologies. Most facial animation editing tools are based on linearly interpolating a set of blendshapes. A skilled animator can achieve an acceptable facial animation result by manually adjusting each blendshape weight one at a time. However, repetitive user interventions related to correcting problematic frames when editing the facial model are necessary until satisfactory results are obtained. Therefore, the demand for an intuitive facial animation editing system that minimizes the time and effort required by an animator to accomplish an editing task has increased significantly.

Direct manipulation approaches allow the user to edit a facial model by pinning and dragging positional constraints that can be arbitrarily defined. Specifically, the direct manipulation framework by Lewis and Anjyo [LA10] determines the blendshape weights by solving a set of linear systems, however, is difficult to successfully explain highly non-linear facial expression spaces. Unintended and

unexpected results can be produced when performing non-linear facial movements in the regions of the lips, chin, and eyelids. Meanwhile, the formulation when editing a facial model via non-linear optimization may produce a plausible result. Unfortunately, an iterative solving process during the non-linear optimization performs slowly such that this strategy is generally unsuitable for real-time applications.

In this paper, we present a novel interactive facial animation editing system that can express the naturalness of non-linear facial movements in real-time. Our method is based on direct manipulation with non-linear blendshape interpolation. We formulate the facial animation editing problem as a two-step optimization problem. First, the blendshape weights are quickly computed from the linearly constrained quadratic optimization [LA10], after which these weights are utilized to solve all facial mesh vertices in the non-linear feature space. MeshIK [SZGP05] simultaneously and iteratively updates the blendshape weights and facial mesh vertices for non-linearly interpolating the deformation gradients of blendshapes. Non-linear optimization will notably incur a computational burden as the number of examples and corresponding vertices increases. Instead of handling such a non-linear system, the proposed method efficiently computes the final mesh vertices only by solving the linear equations once since we reuse the estimated blendshape weights in the first optimization process.

To demonstrate our method quantitatively and qualitatively, we

[†] Equal Contribution

[‡] Corresponding Author

manipulate the facial mesh using different deformation methods with identical scenarios and positional constraints obtained from an optical mocap system. For the positional constraints, a new alignment algorithm between the optical marker positions and the corresponding vertices on the facial mesh is also our technical contribution for accurate performance-driven facial animation. The results show that our method produces the desired and realistic facial animation compared to existing mesh deformation methods that are mainly based on linear combination and optimization.

2. Related Work

For many decades, the blendshape facial rig has been very popular and widely used in many industries to produce realistic facial animations. Blendshape rigs, which consist of semantically separated and meaningful expressions are frequently based on the Facial Action Coding System (FACS) [EF78]. The FACS involves modelling expressions and separating them into different shapes inspired by muscle movements.

Several researchers have explored facial animation editing systems that automatically compute the optimal blendshape weights while meeting certain user constraints. Direct manipulation methods [LA10] allow the user to deform the facial mesh seamlessly by directly pinning and dragging vertices on the facial mesh. Li et al. [LWP10] proposed an automated method that relied on the flip-flop optimization of the blendshape weights and their targets. Yu and Liu [YL14] introduced an approach that modified facial mesh vertices with iterative optimization from captured facial movements. We would like to refer readers to Lewis et al. [LAR*14] for a comprehensive understanding of facial animation systems using blendshapes. These linear model-based approaches are simple and suitable for real-time applications. However, most of them are limited to expressing non-linear facial movements during interactive manipulation. Our method extends direct manipulation in such a way as to integrate the non-linearity into our system nicely by blending the deformation gradients of examples in the non-linear feature space [SZGP05]. In this way, the proposed method can successfully express various non-linear facial movements.

Although facial animation systems based on the deep learning framework [BODO18, BSBS19, RdGM20] have demonstrated impressive results, those are built upon non-public training dataset, which is typically based upon linear-based facial editing and performance capturing. Therefore, unappealing facial animations without non-linear facial movements could be created. Moreover, it would be a notorious task to compile a large amount of facial data to train a neural network faithfully. On the other hand, the proposed method is easy-to-implement and robust. A much smaller set of examples are required when compared to deep learning frameworks.

3. Method

The blendshape model can be defined as a linear combination of predefined facial models via the following equation:

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{B}\mathbf{w}, \quad (1)$$

where $\mathbf{x} \in \mathbb{R}^{3N}$ is the generated facial pose represented in a vector form that consists of all vertex positions of the facial mesh in a

consistent order $xyzxyz\dots$ and $\mathbf{B} \in \mathbb{R}^{3N \times M}$ is a blendshape matrix whose column is a subtraction between the neutral pose and each blendshape. Here, $\mathbf{w} \in \mathbb{R}^M$ is the blendshape weight vector. Manual adjustment of each blendshape weight until the desired results are obtained is a laborious and tedious task.

With regard to the direct manipulation scheme, the user can manipulate the facial mesh by pinning constraints on the surface of the mesh and directly dragging them to new positions. We can compute the pseudo-inverse matrix of $\tilde{\mathbf{B}}$ that denotes the submatrix defined by selecting the rows of \mathbf{B} , then the blendshape weights \mathbf{w} are obtained as $(\tilde{\mathbf{B}}^T \tilde{\mathbf{B}})^{-1} \tilde{\mathbf{B}}^T \mathbf{m}$ where \mathbf{m} is the vector containing the pinning constraints. Unfortunately, the singularity issue may arise when the matrix $\tilde{\mathbf{B}}^T \tilde{\mathbf{B}}$ is not a full rank matrix.

Lewis and Anjyo [LA10] employed ridge regression to address the singularity issue and compute the blendshape weights seamlessly. The weight value of the regularization term controls the trade-off between the accuracy of satisfying the constraints and stability of the blendshape weight. The direct manipulation method is defined as the following quadratic minimization:

$$\mathbf{w}^* = \arg \min_{0 \leq \mathbf{w} \leq 1} \|\tilde{\mathbf{B}}\mathbf{w} - \mathbf{m}\|_2 + \alpha \|\mathbf{w} - \mathbf{w}_0\|_2 + \mu \|\mathbf{w}\|_2, \quad (2)$$

where \mathbf{w} and \mathbf{w}_0 are the current and previous blendshape weights, respectively. The constant value α could help to compute the blendshape weights as closely as possible to the corresponding positional constraints, and μ as a blendshape weight regularization term prevents the weights from abruptly being changed during the editing process. In our experiment, we set α to 0.1 and μ to 0.001. The direct manipulation approach is simple and fast enough as the size of quadratic minimization does not depend on the number of facial mesh vertices, instead depending on the pinning constraints. However, one inherent limitation is that it does not successfully represent non-linear facial movements due to their associated linear formalization.

We formulate the facial animation editing as a two-step optimization problem. The first optimization uses Eq. (2) and compute the blendshape weights. At the second optimization phase, we undertake facial animation editing in the non-linear feature space. Given n facial examples, we can define the n feature vector set $\mathcal{F} = \{\mathbf{f}_l | l = 1, \dots, n\}$ as follows:

$$\mathbf{f} = \{\log \mathbf{R}_{l,i}; \mathbf{S}_{l,i} - \mathbf{I} | \forall t_i \in \mathcal{T}\}, \quad (3)$$

where $\log \mathbf{R}_{l,i}$ is the matrix logarithm representation of rotation matrix for the i -th triangle of the l -th example $\mathbf{R}_{l,i}$, and $\mathbf{S}_{l,i} - \mathbf{I}$ is the subtraction between the scaling/shear part and identity matrix for the i -th triangle of the l -th example. Here, \mathcal{T} denotes the set of triangles on the facial mesh.

Our deformation energy is based on the deformation gradients [SP04, SZGP05]. The deformation gradient of the j -th triangle in the i -th example pose can be defined using the following equation:

$$\mathbf{T}_j(\mathbf{w}) = \exp \left(\sum_{i=1}^l \mathbf{w}_i \log(\mathbf{R}_{i,j}) \right) \cdot \left(\bar{\mathbf{w}}\mathbf{I} + \sum_{i=1}^l \mathbf{w}_i \mathbf{S}_{i,j} \right), \quad (4)$$

where \mathbf{w}_i represents the i -th blendshape weight obtained by solving

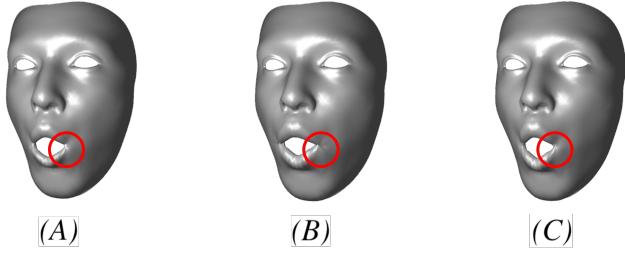


Figure 1: Collision handling. The result from direct manipulation (A). The triangles collide at the corner of the lips without collision handling (B). The collisions are resolved after performing our constrained optimization (C).

Eq. (2), $\bar{\mathbf{w}} = 1 - \sum_{i=1}^l \mathbf{w}_i$, and $\exp(\cdot)$ denotes the matrix exponential operation.

In contrast to the standard non-linear least-squares problem that simultaneously optimizes the weights \mathbf{w} for the examples and face pose \mathbf{x} [SZGP05], we set \mathbf{w} as the blendshape weights in the first optimization (Eq.(2)), and the deformation gradient (Eq.(4)) are concatenated for all triangles. The facial mesh vertices \mathbf{x} are computed by minimizing the equation below:

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \|\mathbf{G}\mathbf{x} - \mathbf{T}(\mathbf{w})\|_2 \quad \text{subject to} \quad \mathbf{C}\mathbf{x} = \mathbf{b}, \quad (5)$$

where \mathbf{G} is sparse and defined from the deformation gradient scheme [SP04, SZGP05]; \mathbf{C} is a diagonal matrix with non-zero entries, and \mathbf{b} is a vector composed of the pinning constraints. By solving Eq. (5), we obtain the final facial mesh vertices $\mathbf{x} = (\mathbf{G}^\top \mathbf{G} + \beta \mathbf{C}^\top \mathbf{C})^{-1} (\mathbf{G}^\top \mathbf{T}(\mathbf{w}) + \beta \mathbf{C}^\top \mathbf{b})$ where β is a controllable parameter for retaining the pinning constraints on the facial mesh. In our experiment, we set β to 100.

During the editing process, self-collisions between triangles can occur and should be resolved (see Fig. 1). As triangles are normally densely packed around the lip region, the corners of the lips are mainly collided while others are resilient to the deformation of the facial mesh. To address this issue, we additionally and automatically impose linear constraints on the vertices of the collided triangle and solve Eq. (5). First, we build the BVH (Bounding Volume Hierarchy) [LG98] that is a tree structure on a set of triangles of the mesh and promptly searches for triangles associated with the collisions. If the distance of a pair of triangles is less than a certain distance threshold ϵ , the vertices of the triangles should be moved to resolve the collisions. As the direct manipulation method barely produces any visible self-collision around the lip region during our performance-driven facial animation (see Fig. 1 (A)), we create the constraints on the vertices penetrating the triangles $\{\mathbf{x}_i\}$ for efficiency and those are represented as a set of equality constraints $\{\mathbf{x}_i = (\mathbf{x}_0 + \mathbf{B}\mathbf{w})_i\}$ where \mathbf{w} is given from Eq. (2) and i indexes the collided vertex. We plug all the equality constraints into Eq. (5) and run the optimization (see Fig. 1 (C)).

For marker-based performance capturing, it is necessary to align the markers obtained from the optical mocap system accurately on

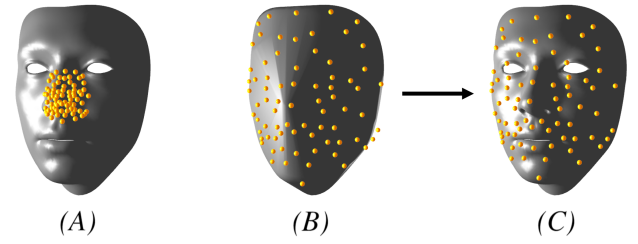


Figure 2: The proposed facial marker alignment. Failure case when a naive ICP computes an incorrect transformation (A). The affine transformed markers ($\hat{\mathbf{q}}$) are well distributed on the coarse facial mesh ($\tilde{\mathbf{p}}$) after running the ICP for the extracted points on the 3D convex hull shape (B). The markers ($\tilde{\mathbf{q}}$) used for the constraints are aligned on the facial mesh (\mathbf{p}) after a two-step ICP with 3D convex hull (C).

Algorithm 1: Facial Marker Alignment

\mathbf{q} : Optical marker points on the neutral face
 \mathbf{p} : Mesh vertices on the neutral face
 $\hat{\mathbf{q}}$: Optical marker points in 3D convex hull
 $\tilde{\mathbf{p}}$: Mesh vertices in 3D convex hull
 $\mathbf{R}, \mathbf{S}, \mathbf{t}$: Affine Transformations for ICP

```

1  $\hat{\mathbf{q}} \leftarrow \text{ConvexHull}(\mathbf{q})$ 
2  $\tilde{\mathbf{p}} \leftarrow \text{ConvexHull}(\mathbf{p})$ 
3  $\{\mathbf{R}, \mathbf{S}, \mathbf{t}\} \leftarrow \text{ICP}(\hat{\mathbf{q}}, \tilde{\mathbf{p}})$  // 1st ICP
4  $\hat{\mathbf{q}} \leftarrow \text{AffineTransform}(\mathbf{q}, \{\mathbf{R}, \mathbf{S}, \mathbf{t}\})$ 
5  $\{\mathbf{R}, \mathbf{S}, \mathbf{t}\} \leftarrow \text{ICP}(\hat{\mathbf{q}}, \mathbf{p})$  // 2nd ICP
6  $\tilde{\mathbf{q}} \leftarrow \text{AffineTransform}(\hat{\mathbf{q}}, \{\mathbf{R}, \mathbf{S}, \mathbf{t}\})$ 
7  $\mathbf{x} \leftarrow (\mathbf{G}^\top \mathbf{G} + \beta \mathbf{C}^\top \mathbf{C})^{-1} (\mathbf{G}^\top \mathbf{T}(\mathbf{w}) + \beta \mathbf{C}^\top \tilde{\mathbf{q}})$ 
    
```

the facial mesh to find the correspondence between them. Therefore, we utilize the ICP (Iterative Closest Point) scheme that is widely used for the point cloud alignment [BM92] and modify this method to deal with the correspondence issue. The ICP is performed to find the optimal transformation between the source $\{\mathbf{q}_i\}$ and target $\{\mathbf{p}_i\}$. However, we may not be able to obtain the desired transformation if the algorithm is directly applied during the process of selecting proper k vertex pair candidates for (\mathbf{q}, \mathbf{p}) , as shown in Fig. 2 (A). To resolve this problem, we propose a two-step ICP method using the 3D convex hull algorithm to extract boundary vertices of the facial mesh (Algorithm 1). First, this method runs the ICP for the marker points and the mesh vertices extracted from the 3D convex hull shape. After applying the affine transformation to the marker points, ICP is then performed again for the transformed markers and facial mesh vertices as shown in Fig. 2 (B). Finally, the affine transformed marker points are used as the pinning constraints to manipulate the facial mesh (see Fig. 2 (C)).

4. Results

We conducted all of the experiments on a publicly available ICT face model [LBZ*20] for editing facial expression. The number of blendshapes in our experiments is 57 and each blendshape con-



Figure 3: (A): Deformation transfer [SP04], (B): Direct manipulation [LA10], (C): Ours. Our method is stable and shows reliable results with realistic shape of the lips and others on specific configurations (i.e., puckering and protruding lips).

sists of 6,706 vertices and 13,120 triangles. We specified the optical mocap markers on the facial model and compared our method with previous works (deformation transfer [SP04], MeshIK [SZGP05], and direct manipulation [LA10]). The performance results were measured on a standard desktop PC with an i7-8700 processor and 32GB of memory. The proposed method animated the examples and ran them completely in real-time at 30 fps. The deformation transfer and direct manipulation methods produced results at 25 fps and 150 fps, respectively. The MeshIK method was too slow for a performance-driven facial animation system due to its performance at 0.03 fps. The computational cost of our system depends on the number of constraints in the first optimization and the number of vertices in the second optimization. We precomputed Eq. (5) by means of Cholesky decomposition and back-substitution to achieve fast performance. Fig. 3 qualitatively compares the proposed method with the others. Deformation transfer [SP04] appeared to work plausible in general, but unexpected shape artifacts could be observed. Direct manipulation [LA10] worked well

in most cases, but the volume and shape of the lower and upper lips were unrealistic during puckering and protruding lips.

5. Conclusion

We have presented a novel interactive facial expression editing system with non-linear blendshape interpolation. One of the key techniques of our approach is sequentially optimizing the blendshape weights and facial mesh vertices. The combination of direct manipulation and non-linear interpolation of the examples effectively produces facial expressions with various and natural facial appearances compared to previous methods. One limitation of our system is that the excessive manipulation could result in crumpling artifacts. For future work, we have a plan to create 3D facial animation from 2D video image sequences in real-time.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (NRF-2019R1F1A1063467). The mocap data were supported by NC.

References

- [BM92] BESL P., MCKAY N. D.: A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14, 2 (1992), 239–256. doi:10.1109/34.121791. 3
- [BODO18] BAILEY S. W., OTTE D., DILORENZO P., O'BRIEN J. F.: Fast and deep deformation approximations. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12. 2
- [BSBS19] BERSON E., SOLADIE C., BARRIELLE V., STOIBER N.: A robust interactive facial animation editing system. In *Motion, Interaction and Games*. 2019, pp. 1–10. 2
- [EF78] EKMAN P., FRIESEN W. V.: Facial action coding system. *Environmental Psychology & Nonverbal Behavior* (1978). 2
- [LA10] LEWIS J. P., ANJYO K.-I.: Direct manipulation blendshapes. *IEEE Computer Graphics and Applications* 30, 4 (2010), 42–50. 1, 2, 4
- [LAR*14] LEWIS J. P., ANJYO K., RHEE T., ZHANG M., PIGHIN F. H., DENG Z.: Practice and theory of blendshape facial models. *Eurographics (State of the Art Reports) 1*, 8 (2014), 2. 2
- [LBZ*20] LI R., BLADIN K., ZHAO Y., CHINARA C., INGRAHAM O., XIANG P., REN X., PRASAD P., KISHORE B., XING J., LI H.: Learning formation of physically-based face attributes, 2020. arXiv: 2004.03458. 3
- [LG98] LIN M., GOTTSCHALK S.: Collision detection between geometric models: A survey. In *Proc. of IMA conference on mathematics of surfaces* (1998), vol. 1, Citeseer, pp. 602–608. 3
- [LWP10] LI H., WEISE T., PAULY M.: Example-based facial rigging. *Acm transactions on graphics (tog)* 29, 4 (2010), 1–6. 2
- [RdGM20] RADZHOVSKY S., DE GOES F., MEYER M.: Facebaker: Baking character facial rigs with machine learning. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks* (2020), pp. 1–2. 2
- [SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)* 23, 3 (2004), 399–405. 2, 3, 4
- [SZGP05] SUMNER R. W., ZWICKER M., GOTSCHALK S., POPOVIĆ J.: Mesh-based inverse kinematics. *ACM transactions on graphics (TOG)* 24, 3 (2005), 488–495. 1, 2, 3, 4
- [YL14] YU H., LIU H.: Regression-based facial expression optimization. *IEEE Transactions on Human-Machine Systems* 44, 3 (2014), 386–394. 2