Accurate Face Rig Approximation with Deep Differential Subspace Reconstruction

STEVEN L. SONG*, Blue Sky Studios
WEIQI SHI*, Yale University
MICHAEL REED, Blue Sky Studios

Fig. 1. Our rig approximation method learns localized shape information in differential coordinates and, separately, a subspace for mesh reconstruction.

To be suitable for film-quality animation, rigs for character deformation must fulfill a broad set of requirements. They must be able to create highly stylized deformation, allow a wide variety of controls to permit artistic freedom, and accurately reflect the design intent. Facial deformation is especially challenging due to its nonlinearity with respect to the animation controls and its additional precision requirements, which often leads to highly complex face rigs that are not generalizable to other characters. This lack of generality creates a need for approximation methods that encode the deformation in simpler structures. We propose a rig approximation method that addresses these issues by learning localized shape information in differential coordinates and, separately, a subspace for mesh reconstruction. The use of differential coordinates produces a smooth distribution of errors in the resulting deformed surface, while the learned subspace provides constraints that reduce the low frequency error in the reconstruction. Our method can reconstruct both face and body deformations with high fidelity and does not require a set of well-posed animation examples, as we demonstrate with a variety of production characters.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Animation**.

Additional Key Words and Phrases: rigging, deep learning, facial animation

ACM Reference Format:
Steven L. Song, Weiqi Shi, and Michael Reed. 2020. Accurate Face Rig Approximation with Deep Differential Subspace Reconstruction. *ACM Trans. Graph.* 39, 4, Article 34 (July 2020), 12 pages. https://doi.org/10.1145/3386569.3392491

1 INTRODUCTION

Film-quality character rigs rely on a complex hierarchy of procedural deformers, driven by a large number of animation controls, that map to the deformation of the vertices of a character's surface mesh. Because the characters are subject to high aesthetic standards, and the rigs are the primary means by which the animators interact with them, the rigs themselves have strict performance requirements: the character's skin must behave predictably and precisely over the entire range of control, which for animated characters can be extreme because of the caricatured design and motion.

*Authors contributed equally.

Authors' addresses: Steven L. Song, stevens@blueskystudios.com, Blue Sky Studios, 1 American Ln, Greenwich, CT, 06831; Weiqi Shi, weiqi.shi@yale.edu, Yale University, New Haven, CT, 06520; Michael Reed, reed@blueskystudios.com, Blue Sky Studios, 1 American Ln, Greenwich, CT, 06831.

0730-0301/2020/7-ART34 $15.00
https://doi.org/10.1145/3386569.3392491

Rigs for facial animation typically have much more complex behavior than body rigs, and require additional precision due to their importance in conveying the most crucial aspects of communication and expression. To offer artistic freedom, the face rig is usually a complex structure containing a large number of numerical controls. Unlike the joint-based controls commonly used for a character's body, these numerical controls are globally defined and cooperatively influence the transformation of each vertex, making facial deformation highly nonlinear and expensive to compute.

In production it's often desirable to reuse the same rig behavior for different purposes in different environments. For example, transferring the rig to a simulation application for crowd simulation, to a game engine for VR production, or to a renderer for render-time manipulation. Unfortunately it's often not viable to take the original rig to other packages because a visually-matching reimplementation is required per deformer per package. Similarly, simulation-based rigs (e.g. muscle systems) provide complex behavior that is desirable in many production situations, but their lack of interactive response discourages their adoption. These issues can be addressed by a rig approximation method if it has the following characteristics: simple universal structure, high accuracy and good performance. A neural network approach automatically meets the first requirement, as the same network can approximate varying non-linear functions with different sets of weights. Neural networks can also provide benefits with batch evaluation. For example crowd characters, which can often reuse the same nonlinear deformation with different scaling factors, can be batch evaluated if driven by a neural network. Much of the work in this area – on moving from the typical rig deformer "stack" to a neural representation – has focused on run-time performance e.g. [Bailey et al. 2018].

In contrast, our work directly addresses the importance of accuracy as experienced in the film production environment. In this paper we introduce a new learning-based solution to accurately capture facial deformation for characters using differential coordinates and a network architecture designed for that space. Similar to other work, we assume that the deformation has both a linear and a nonlinear component that can be separated. The linear deformation is not the focus of this paper since its contribution to facial deformation is limited and many linear skinning solutions have been proposed [Kavan et al. 2008; Kavan and Žára 2005]. Instead, we focus on learning the nonlinear component, which applies equally well to both face and body rig approximation, as we show in our results.

At run-time our method takes as input animation controls defined as a set of artist-level rig parameters, and computes the deformation as vertex displacements from the rest pose. During the offline training process, we use vectorized features generated from rig parameters, and labels are differential coordinates calculated from the localized nonlinear deformation of the original rig. The differential coordinates have the advantages of a sparse mesh representation and embedded neighbor vertex information, which contribute to the learning of local surface deformation. However, the transformation between coordinates is ill-conditioned and non-invertible, and so we introduce a separate subspace to improve the conditioning of the reconstruction. This subspace is determined by artist-specified "anchor points", selected from the original mesh at features that are significant to the character's expressive ability. Our method conducts separate subspace training to learn how these anchor points deform using a split network structure.

We qualitatively and quantitatively evaluate our method on multiple production-quality facial rigs. Experimental results show our method can predict accurate facial deformation with minimal visual difference from the ground truth. We show our method extends to body deformation where it compares favorably with existing solutions. Additionally, we show how using anchor points improves the reconstruction by reducing the low frequency error introduced in the differential training.

2 RELATED WORK

2.1 Skinning and Rigging

Skinning techniques can be roughly divided into physics-based [Kim et al. 2017; Si et al. 2014], example-based [Loper et al. 2015; Mukai and Kuriyama 2016], and geometry-based methods. We focus here on geometry-based solutions due to their computational efficiency and simplicity. One of the most widely used techniques is linear blend skinning (LBS) [Magnenat-Thalmann et al. 1988], where a weighted sum of the skeleton's bone transformations is applied to each vertex. Advances in this technique include dual quaternion skinning (DQS) [Kavan et al. 2008], spherical blend skinning [Kavan and Žára 2005] and optimized centers of rotation skinning [Le and Hodgins 2016]. Although these methods are computationally efficient for computing linear deformation, they do not handle nonlinear behaviors such as muscle bulging and twisting effects. Improving on this, Merry et al. [2006] and Wang et al. [2002] introduce more degrees of freedom for each bone transformation through additional skin weights, which can be acquired by fitting example poses. Other approaches designed to address these issues include pose space deformation [Lewis et al. 2000; Sloan et al. 2001], cage deformation [Jacobson et al. 2011; Joshi et al. 2007; Ju et al. 2005; Lipman et al. 2008], joint-based deformers [Kavan and Sorkine 2012], delta mush [Le and Lewis 2019; Mancewicz et al. 2014] and virtual/helper joints methods [Kavan et al. 2009; Mukai 2015; Mukai and Kuriyama 2016]. Wang et al. [Wang et al. 2007] introduce a rotational regression model to capture nonlinear skinning deformation, which optimizes the deformation of all vertices simultaneously using the Laplace equation. An iterative optimization [Sorkine and Alexa 2007] is proposed to approximate nonlinear deformation by alternating surface smoothing and local deformation. All of these methods require additional computational cost for nonlinear components and are primarily focused on body deformation, leaving facial deformation largely unaddressed.

2.2 Facial Rig and Deformation

In contrast to body rigs that are defined by bones and joints, facial rigs often include hundreds of animation controls represented by numerical values which control the nonlinear transformation of each vertex. These animation controls are globally defined and widely used in blendshapes [Lewis et al. 2014; Lewis and Anjyo 2010] to achieve realistic facial animation for production. Prior work focused on editing data-driven facial animation [Deng et al. 2006; Joshi et al. 2006] or providing intuitive control [Lau et al. 2009; Lewis and Anjyo

2010]. Li et al. [Li et al. 2010] successfully transfer controller semantics and expression dynamics from a generic template to the target model using blendshape optimization in gradient space. Weise et al. [Weise et al. 2011] present a high-quality performance-driven facial animation system for capturing facial expressions and creating a digital avatar in real-time. A blendshape system that allows efficient anatomical and biomechanical facial muscle simulation is proposed in [Cong et al. 2016].

2.3 Learning-based Deformation

There has been increasing interest in using learning-based solutions to replace traditional deformation algorithms. Previous work such as [Lewis et al. 2000] utilize a support vector machine to learn mesh deformation given a set of poses. [Tan et al. 2018a,b] propose mesh-based autoencoders to learn deformation from a latent space. Based on their work, [Gao et al. 2018] put forward a solution to transfer shape deformation between characters with different topologies using a generative adversarial network. Luo et al. [2018] propose a deep neural network solution to approximate nonlinear elastic deformation, combining this with simulated linear elastic deformation to achieve better results. Liu et al.[2019] use graph convolutional networks to predict the skin weight distribution for each vertex, resulting in a trained network that can be applied to different characters given their mesh data and rigs. Relevant to our work is [Bailey et al. 2018], where multiple neural networks are used to approximate the rig's nonlinear deformation components under the assumption that each vertex is associated with a single bone. For each bone, they train a network to predict the offset of each associated vertex. Three unaddressed issues that motivate our work are: (1) the deformation of a vertex is often influenced by multiple bones, with no single bone as the prominent influence, (2) the deformation can be determined by numeric controls (as in face rigs) and (3) associating bones with disjoint sets of vertices can introduce discontinuities at set boundaries.

2.4 Subspace Deformation and Model Reduction

Subspace model reduction techniques are commonly used to solve nonlinear deformation in real-time applications. Instead of evaluating the complete mesh, subspace models compute the deformation of a low dimensional embedding on the fly and project it back to the entire space. Subspace deformation was originally used in early simulation work [Pentland and Williams 1989], which uses a subspace spanned by the low-frequency linear vibration modes to represent the deformation. To augment the linear model and handle nonlinearities, Krysl et al. [2001] propose the empirical eigenvectors subspaces using principal component analysis (PCA) for finite element models. Summer et al. [2007] use graph structure to represent deformations as a collection of affine transformations for shape manipulation. An et al. [2008] introduces subspace forces and Jacobians associated with subspace deformations for simulation. Barbič et al. [2005] observe that the reduced internal forces with linear materials are cubic polynomials in reduced coordinates, which could be precomputed for efficient implicit Newmark subspace integration. For deformation-related model reduction, Barbič et al. [2012] propose a method for interactive editing and design of deformable object animations by minimizing the force residual objective. Wang et al. [2015] design linear deformation subspaces by minimizing a quadratic deformation energy to efficiently unify linear blend skinning and generalized barycentric coordinates. Building on these works, a recent hyper-reduced scheme [Brandt et al. 2018] uses two subspaces to achieve real-time simulation, one for constraint projections in the preprocessing stage and the other for vertex positions in real-time. Close to our work is Meyer et al. [2007], who propose the Key-Point Subspace Acceleration (KPSA) and caching to accelerate the posing of deformable facial models. The idea of using key points for reconstruction is analogous to the anchor points in our case. However, their method, like other subspace techniques, relies on high quality animation prior examples to compute the embedding of the subspace.

Compared with previous work, the advantages of our method are: (1) it can reconstruct both face and body deformation with high accuracy, (2) it can take different types of animation controls as input, (3) it does not require a particular set of well-posed animation priors and (4) it provides a simple universal structure for cross-platform real-time evaluation.

For the rest of the paper, we first review the preliminaries of differential coordinates in Section 3.1. We then describe our training pipeline (Section 3.2), including the vectorization from input animation controls, the acquisition of nonlinear deformation from existing poses, network structures and reconstruction. We introduce the implementation details in Section 3.3, and we describe our experiments, evaluate the training results, compare with existing solutions in Section 4. Finally, Section 5 discusses limitations and future work.

3 METHOD

Our model approximates the nonlinear deformation in a character rig. The linear deformation can be simply represented with linear blend skinning, so it's not our focus here. For a given mesh in rest pose, our model takes animation controls defined by a set of rig parameters as inputs, and outputs the non-linear deformation of the mesh. Fig. 2 shows our training pipeline. To process the training data, we first vectorize the input rig parameters and extract the nonlinear deformation represented by vertex displacement from the corresponding deformed mesh. Then we convert the nonlinear deformation into differential coordinates (δ space), where we learn localized shape information and map the rig controls to it. However, we cannot directly reconstruct the mesh surface from differential coordinates since the transformation is ill-conditioned. We conduct a separate subspace learning on a group of anchor points selected from the original mesh, for which we learn deformation in local coordinates and use them as constraints for reconstruction.

3.1 Preliminary

Let $\boldsymbol{M} \in \{\boldsymbol{V}, \boldsymbol{E}\}$ be a mesh with n vertices, $\boldsymbol{V} \in \mathbb{R}^{n\times 3}$. Each vertex $\boldsymbol{v}_i \in \boldsymbol{V}$ is represented using absolute Cartesian coordinates and $\boldsymbol{E}$ represents the set of edges. The Laplacian operator $\boldsymbol{L}$ is defined [Sorkine 2005] as:

$$\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A} \tag{1}$$

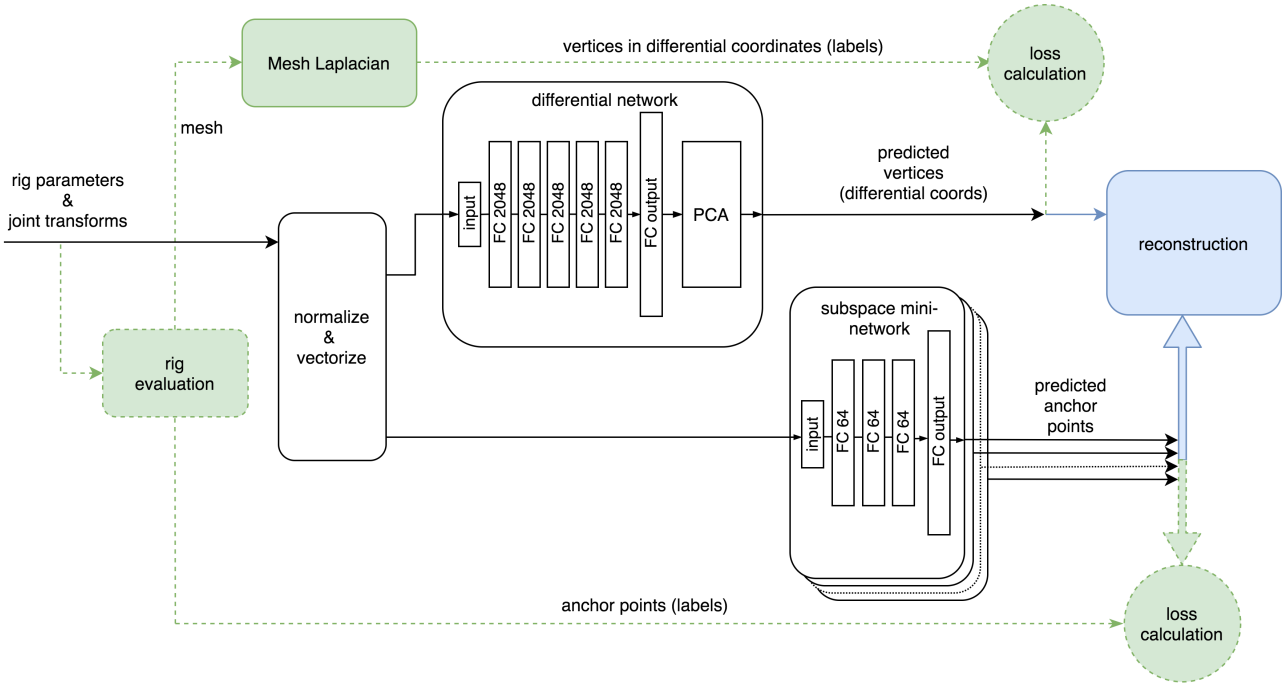

Fig. 2. Our method takes rig parameters and the corresponding joint transforms as input and predicts the nonlinear deformation of the mesh vertices (in differential coordinates) and the set of anchor points (in cartesian space). Green pathways are for network training, blue pathways for prediction.

where $\boldsymbol{A}$ is a (0, 1) adjacent matrix of size $n \times n$ that indicates the connectivity of vertex pairs in the mesh with $\boldsymbol{A}_{ij} = 1$ if $(i, j) \in \boldsymbol{E}$. $\boldsymbol{D}$ is a diagonal matrix of size $n \times n$ representing the degree d_i of each vertex. Applying the Laplacian operator $\boldsymbol{L}$ to the vertices transforms the mesh into delta space, where each vertex $\boldsymbol{v}_i$ is represented as $\boldsymbol{\delta}_i$. The differential coordinate of each vertex represents the difference between the vertex itself and the center of mass of its immediate neighbors (A_i denotes the neighborhood set of vertex $\boldsymbol{v}_i \in \boldsymbol{V}$):

$$\begin{aligned} \boldsymbol{LV} &= \boldsymbol{\delta} \\ \boldsymbol{v}_i - \frac{1}{d_i} \sum_{j \in A_i} \boldsymbol{v}_j &= \delta_i \end{aligned} \tag{2}$$

It's more convenient to use the symmetrical version of $\boldsymbol{L}$, denoted by $\boldsymbol{L_s} = \boldsymbol{DL} = \boldsymbol{D} - \boldsymbol{A}$, giving:

$$\boldsymbol{L_s V} = \boldsymbol{D\delta} \tag{3}$$

Compared to the Cartesian coordinates, where only the spatial location of each vertex is provided, the differential coordinates carry information about the local shape of the surface and the orientation of local details. It preserves local surface detail and captures the irregular shape of the surface. Transferring mesh deformation data into differential space leads to a sparse representation, which also contributes to the learning process. Intuitively, if a surface patch is deformed uniformly, the differential representation of the deformation will have zero values for all vertices except for the boundaries.

Given the Laplacian operator and differential coordinates, we now consider how to reconstruct mesh surface. Note the matrix $\boldsymbol{L_s}$ is singular and has a non-trivial zero eigenvector because the sum of all its rows is 0. Therefore, we cannot directly invert the matrix for reconstruction, but can add constraints to the matrix to make it full rank. We introduce the subspace $\boldsymbol{P}$, which is constructed by a set of anchor points from $\boldsymbol{V}$. The dimension of the subspace is much smaller than the original mesh. The index matrix of the anchor points $I(\boldsymbol{P})$ is appended at the end of the Laplacian matrix $\boldsymbol{L_s}$. Correspondingly, we append the Cartesian coordinates of anchor points $\boldsymbol{V}(\boldsymbol{P})$ to the differential coordinates of the full mesh to make it solvable:

$$\widetilde{\boldsymbol{L}}\boldsymbol{V} = \begin{pmatrix} \boldsymbol{L_s} \\ \boldsymbol{\omega} I(\boldsymbol{P}) \end{pmatrix} \boldsymbol{V} = \begin{pmatrix} \boldsymbol{D\delta} \\ \boldsymbol{\omega V}(\boldsymbol{P}) \end{pmatrix} = \widetilde{\boldsymbol{\delta}} \tag{4}$$

$\widetilde{\boldsymbol{L}}$ is the full-rank matrix with anchor points appended to the original Laplacian matrix. $\boldsymbol{\omega}$ is the weight matrix for the anchor points, which can be used to stress the importance of each anchor points. Given the full rank matrix $\widetilde{\boldsymbol{L}}$ and $\widetilde{\boldsymbol{\delta}}$, we can solve the following equation:

$$(\widetilde{\boldsymbol{L}}^T \widetilde{\boldsymbol{L}})\boldsymbol{V} = \widetilde{\boldsymbol{L}}^T \widetilde{\boldsymbol{\delta}} \tag{5}$$

Applying the Laplacian operator to a mesh is analogous to obtaining the second spatial derivatives. The eigenvectors of $\boldsymbol{L}$ are cosine basis functions of the Fourier transform, and the associated eigenvalues are squares of the frequencies [Zhang et al. 2010]. We demonstrate that for a small error $\boldsymbol{\epsilon}$ introduced in differential coordinates, the high frequency component of $\boldsymbol{\epsilon}$ is dampened when

converted back to Cartesian space. This leads to a smoother distribution of the error, which is much less noticeable in the reconstructed surface.

Since $\boldsymbol{L_s}$ is symmetric positive semi-definite, it has an orthogonal eigenbasis $\boldsymbol{E} = \{\boldsymbol{e_1}, \boldsymbol{e_2}, ...\boldsymbol{e_n}\}$, with corresponding eigenvalues $0 < \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq ... \leq \lambda_n$. (For this analysis, we assume $\boldsymbol{L_s}$ is non-singular by adding one anchor)

$$\begin{aligned} \boldsymbol{L_s V'} &= \boldsymbol{D(\delta + \epsilon)} \\ \boldsymbol{V'} &= \boldsymbol{L_s^{-1} D(\delta + \epsilon)} \\ \boldsymbol{V'} &= \boldsymbol{V + L_s^{-1} D\epsilon} \end{aligned} \tag{6}$$

We denote $\boldsymbol{D\epsilon}$ as $\boldsymbol{\epsilon'}$ and decompose it in basis $\boldsymbol{E}$

$$\boldsymbol{\epsilon'} = c_1\boldsymbol{e_1} + c_2\boldsymbol{e_2} + ...c_n\boldsymbol{e_n} \tag{7}$$

Notice that $\boldsymbol{L_s}^{-1}$ shares the same eigenvectors and its corresponding eigenvalues are inversed. We have

$$\boldsymbol{L_s}^{-1}\boldsymbol{\epsilon'} = \frac{1}{\lambda_1}c_1\boldsymbol{e_1} + \frac{1}{\lambda_2}c_2\boldsymbol{e_2} + ...\frac{1}{\lambda_n}c_n\boldsymbol{e_n} \tag{8}$$

Since λ_1 is small and λ_n is large, the inverse of the eigenvalues amplifies the low frequency eigenvector $\boldsymbol{e_1}$ and dampens the high frequency one $\boldsymbol{e_n}$. In this way, the high-frequency errors in the differential coordinates are reduced. This is desirable for mesh deformation as localized high frequency errors are much more noticeable. To reduce the amplification of low-frequency error, we increase the number of anchor points, which improves the conditioning of the Laplacian matrix by increasing the smallest singular value. Therefore, we can decrease both the low and high-frequency errors when the mesh surface is reconstructed.

3.2 Pipeline

3.2.1 Input Features. The rig parameters cannot be directly used for training because they are in different representations and scales. Therefore, we need to first create feature vectors from the given rig parameters. Without loss of generality, we assume that facial rigs include joint controls $\boldsymbol{J}$ and numerical controls $\boldsymbol{C}$. For the joint controls, we use the transformation matrix $\boldsymbol{M_{J_i}} = [\boldsymbol{X_{J_i}}, \boldsymbol{t_{J_i}}]$ of each joint $\boldsymbol{J_i}$ as input, where $\boldsymbol{X_i} \in \mathbb{R}^{3\times3}$ is the rotation/scale matrix and $\boldsymbol{t_i} \in \mathbb{R}^3$ is the normalized translation value. We vectorize and concatenate all the joint controls so that we have $\boldsymbol{J} = \{J_1, ...J_i, ...J_j\}, J_i \in \mathbb{R}^{12}$. For the numerical controls, we define the input features as the concatenation of the normalized numerical value of each attribute, $\boldsymbol{C} = \{C_1, ...C_i, ...C_c\}, C_i \in \mathbb{R}^1$, where C_i represents each control attribute. Then we concatenate all the joint and numerical controls as our input feature $\boldsymbol{F}$, whose dimension is $12j + c$. We normalize all the translation values together, but every single numerical control attribute is normalized independently since they are on different scales.

$$\boldsymbol{F} = Concat(||_{i=1}^{j}\boldsymbol{J_i}, ||_{i=1}^{c}C_i) \tag{9}$$

To generate the training data, we randomly and independently sample each rig control using truncated Gaussian distribution within a set range. The range of each control is defined so that it reasonably covers the possible range of animation, similar to the method used by [Bailey et al. 2018]. We do not limit our training data to well-animated poses because (1) they require human labor and thus are expensive to generate, and (2) using randomly generated poses can cover a large range of motion and more dynamic deformations, which can improve the generalization of our model.

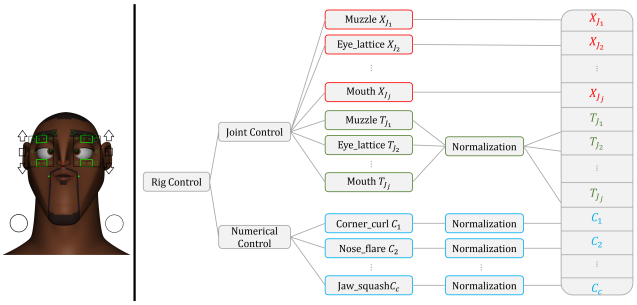

Fig. 3. An example for rig controls and vectorization. Only joint controls are shown on the character.

3.2.2 Nonlinear Deformation. We use the nonlinear deformation as our training labels, which can be computed from the deformed mesh. We assume a mesh in rest pose $\boldsymbol{V}$ and its deformation $\widetilde{\boldsymbol{V}}$ is defined by a set of rig parameters. We also assume $\widetilde{\boldsymbol{V}}$ and $\boldsymbol{V}$ maintain the same topology. The vertex $\boldsymbol{v_i} \in \boldsymbol{V}$ and $\widetilde{\boldsymbol{v}}_i \in \widetilde{\boldsymbol{V}}$ are defined in local Cartesian coordinates. We have the following equation:

$$\widetilde{\boldsymbol{v}}_i = \boldsymbol{T_i}(\boldsymbol{v_i} + \boldsymbol{v}_{i,nl}) \tag{10}$$

where $\boldsymbol{v}_{i,nl}$ is the vertex displacement in local space caused by the nonlinear deformation. $\boldsymbol{T_i}$ is the linear transformation for vertex $\boldsymbol{v_i}$ which can be computed from the transformation matrix of the joint controls.

$$\boldsymbol{T_i} = \sum_{k=1}^{\boldsymbol{J}(\boldsymbol{v_i})} \omega_k \boldsymbol{M_{J_k}} (\boldsymbol{M^o_{J_k}})^{-1} \tag{11}$$

$\boldsymbol{J}(\boldsymbol{v_i})$ represents the joint controls that have influence on the vertex $\boldsymbol{v_i}$. $\boldsymbol{M_{J_k}}$ denotes the transformation matrix for joint $\boldsymbol{J_k}$ and $\boldsymbol{M^o_{J_k}}$ is its transformation matrix at rest pose. ω_k is the weight for the joint. We assume the rig as a black box, so we don't have $\boldsymbol{M^o_{J_k}}$ and ω_k available. For general purposes, we use an implicit method to calculate $\boldsymbol{T_i}$. Given equation 10, we perturb $\boldsymbol{v}_{i,nl}$ by moving one unit for every direction along XYZ coordinates and observe the vertex displacement produced by the rig. Then we can use the vertex displacement to calculate $\boldsymbol{T_i}$. With the following equations:

$$\begin{aligned} \widetilde{\boldsymbol{v}}'_i &= \boldsymbol{T_i v_i} \\ \widetilde{\boldsymbol{v}}_{i,x} &= \boldsymbol{T_i}(\boldsymbol{v_i} + (1,0,0,0)^T) \\ \widetilde{\boldsymbol{v}}_{i,y} &= \boldsymbol{T_i}(\boldsymbol{v_i} + (0,1,0,0)^T) \\ \widetilde{\boldsymbol{v}}_{i,z} &= \boldsymbol{T_i}(\boldsymbol{v_i} + (0,0,1,0)^T) \\ \widetilde{\boldsymbol{v}}_{null} &= \boldsymbol{T_i}(0,0,0,1)^T \end{aligned} \tag{12}$$

By subtracting the first equation from the following ones, we have:

$$\boldsymbol{T_i} = (\widetilde{\boldsymbol{v}}_{i,x} - \widetilde{\boldsymbol{v}}'_i, \quad \widetilde{\boldsymbol{v}}_{i,y} - \widetilde{\boldsymbol{v}}'_i, \quad \widetilde{\boldsymbol{v}}_{i,z} - \widetilde{\boldsymbol{v}}'_i, \quad \widetilde{\boldsymbol{v}}_{null}) \tag{13}$$

$\boldsymbol{T}_i$ can be substituted into equation 10 to calculate the nonlinear deformation with given rig input:

$$\boldsymbol{v}_{i,nl} = \boldsymbol{T}_i^{-1}\widetilde{\boldsymbol{v}}_i - \boldsymbol{v}_i \tag{14}$$

Our goal is to learn the nonlinear deformation from given rig parameters by minimizing the per-vertex distance between our results and the ground truth.

3.2.3 Differential Network. The differential network takes the vectorized features as input and outputs the vertex displacement corresponding to the nonlinear deformation in differential coordinates. This network has 5 fully connected layers with 2048 units, each followed by a Relu activation layer. Similar to [Bailey et al. 2018; Laine et al. 2017], we apply PCA at the end of the network by multiplying the projection matrix with the output. We precompute the projection matrix on the entire training set. The training data can be constructed as a matrix $\boldsymbol{M} \in \mathbb{R}^{3|V|\times m}$ where $|V|$ is the vertex count and m is the dimension for all training poses. The purpose of PCA is to project the network output back to a lower dimension which helps the network converge. We determine the number of principal components as a fixed percentage of the number of mesh vertices, which is simple to implement in practice (we evaluate the influence of different percentage on training in Section 4.1). Alternatively the PC number can be selected by choosing the most significant basis vectors such that the reprojection error of the training set is below a defined threshold.

For the loss function, a simple choice would be the regression loss such as the Euclidean distance between the predicted vertex displacement and the ground truth. However, it is known that an L2 loss function tends to blur the prediction results [Isola et al. 2017; Liu et al. 2019]. The mesh deformation for character animation is smooth and continuous, which implies the differential representation has small values. Our training data is generated by random sampling the rig parameters, but this also means the training data contains outliers that would never appear in real animation and which appear in delta space as large values. L2 loss is more sensitive to outliers due to the consideration of the squared differences. In our case, L2 loss tends to adjust the network to fit and minimize those outlier vertices, which leads to higher errors for other vertices. On the other hand, using L1 loss reduces the influence of outliers and produces better result. Therefore we use the L1 loss for the differential network.

3.2.4 Subspace Network. The subspace network takes the vectorized features as input and outputs the nonlinear deformation of selected anchor points in local Cartesian coordinates for reconstruction. Previously, Chen et al. [2005] and Sorkine et al. [2005] use greedy heuristic methods to select anchor points. They treat all the vertices in the mesh equally and iteratively select the vertex based on the largest geodesic distance between the approximated shape and the original mesh. However, these algorithms do not fit in our situation because of the different contributions of vertices to the facial animation. We pay more attention to the important facial features, such as eyes and mouth, rather than nose, ears or the scalp. In general face rigs define the controls on those areas to constrain the deformation. Therefore, we use the rig as reference to select anchor points and make sure that they are well-distributed and proportional to the density of the rig controls. Based on our observation, the training performance and reconstruction results do not depend on the specific anchor point selection as long as the major deformable facial features are covered. We also note that the number of anchor points contributes to the accuracy of reconstruction; we evaluate that in Section 4.2.

The subspace network consists of a set of mini-networks, each of which corresponds to a single anchor point and outputs its deformation in $\mathbb{R}^3$. For the input of each mini-network, we perform a dimension reduction technique similar to that used in [Bailey et al. 2018], where each network takes as input a subset of the vectorized features corresponding to the rig controls that deform the anchor point. However, the difference between our method and Bailey et al. is that we perform the split training on the anchor points instead of the entire mesh, and so we avoid the discontinuity issue. We apply this technique because only a small subset of all rig controls influence a certain anchor point. We collect the related rig controls by perturbing all the controls individually and recording which anchor produces deformation. This process is repeated with 100 random example poses and with large perturbations to ensure that controls affecting the anchor are identified. Each mini-network includes 3 fully connected layers with 64 units, each followed by a Relu activation layer. For the loss function, we use L2 loss for the network as the subnetwork is trained on Cartesian coordinates, which don't encode mesh information in a way that accentuates outliers. We use multiple mini-networks instead of a single network because there is no direct spatial relationship between the anchor points and there is low correlation between their deformation. In practice, we found this structure has better training performance compared with the single network due to the reduced dimension.

3.2.5 Reconstruction. We perform reconstruction using the full-rank Laplacian matrix $\widetilde{\boldsymbol{L}}$, which is constructed by appending the indices of anchor points at the end of the original Laplacian matrix $\boldsymbol{L}$. Notice $\widetilde{\boldsymbol{L}}$ does not vary with input rig parameters and only depends on the selected anchor points. According to equation 5, we can apply Cholesky factorization on $\widetilde{\boldsymbol{L}}^T\widetilde{\boldsymbol{L}}$ to get the upper-triangular sparse matrix $\boldsymbol{R}$:

$$\widetilde{\boldsymbol{L}}^T\widetilde{\boldsymbol{L}} = \boldsymbol{R}^T\boldsymbol{R} \tag{15}$$

We only need to compute the factorization once with only the mesh topology information and the matrix $\boldsymbol{R}$ can be reused whenever rig parameters change. Now we can easily solve the equation 4 and reconstruct the mesh surface using back substitution. We concatenate the results from the differential and subspace network to get $\widetilde{\boldsymbol{\delta}}$ and use it in the following equation:

$$\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{V}_{\boldsymbol{nl}} = \widetilde{\boldsymbol{L}}^T\widetilde{\boldsymbol{\delta}} \tag{16}$$

Since $\boldsymbol{R}$ is a triangular matrix, we can efficiently reconstruct the nonlinear deformation $\boldsymbol{V}_{\boldsymbol{nl}}$ with back substitution, which makes it possible to run the reconstruction at an interactive speed with frequently updated results from the networks.

We use uniform Laplacian instead of the cotangent Laplacian because the latter changes as the mesh deforms, requiring expensive

Table 1. Statistics for the three test models.

	Agent	Bull	Matador
Vertices	4403	3669	3211
Face Height (cm)	25.12	84.28	26.03
Face Width (cm)	21.27	67.00	20.45
Numerical Controls	67	131	121
Joint Controls	20	20	20
Anchor	87	73	64
Differential PC	220	183	160

recomputation for every pose. With uniform Laplacian the factorization only needs to happen once, and the reconstruction is done with 2 back substitutions, which are very fast.

3.3 Implementation Details

For both the differential and subspace networks, we set the batch size as 128 and choose a SGD solver for optimization with the initial learning rate as 0.1 and the learning rate decay as 10^{-6} (SGD outperforms Adam in our case). We train 10000 epochs for both the network, which takes 3.5 hours for the differential network on an NVIDIA GeForce GTX 2080 GPU, and less than 1 hour for the subspace network.

4 EVALUATION

We use three production face rigs for experiments and evaluation (see Table 1). For each rig, we take a truncated normal sampling of the rig parameters to generate 10000 random poses: 9800 for training and 200 for testing (Fig. 4). The test poses are separated from the training data to avoid bias. The rig parameters of the test poses are fed into the trained network to produce the reconstructed deformation (Fig. 2). To evaluate the training performance we use two metrics: the MSE of the prediction error and the reconstruction error (*cm*). The MSE of the prediction error measures the difference between the ground truth and network output, while the reconstruction error measures the per-vertex absolute distance between the surface reconstruction and ground truth deformation. We evaluate the mean and maximum reconstruction errors calculated from the vertices among all test poses. The maximum error is a critical value to consider as a large localized error will render the animation pose unacceptable, regardless of the MSE.

Because face rigs precisely control the eyelid, eyebrow, and mouth behavior, and because these are the primary cues for expression, having high accuracy here is paramount. A slight difference in eyelid position changes the relative position of the pupil, which can change the audience perception of the pose from "scheming" to "sleepy", while a similar change in the lip position can go from "slight smile" - with the teeth slightly exposed - to "sneer", making any method that could not accurately differentiate between these poses unacceptable.

4.1 Evaluation for Differential Training

We first evaluate how varying the number of principle components (PC) influences the differential training. We specify the PC number as a varying percentage of the mesh vertex count. Fig. 5 shows the prediction error for three characters over 200 test poses. It is interesting to note that their MSE is minimized as the PC percentage approaches 5%, regardless of the different number of vertices in their meshes. This suggests the optimal PC number is roughly proportional to the mesh vertex count. Further increasing the PC percentage does not lead to significant performance improvement, but instead makes the network vulnerable to overfitting, shown by the slight increasing of the loss. Based on these observations, we set the PC number as 5% of the mesh vertex count for differential training for the rest of our evaluation.

Table 2. Prediction error (differential) and reconstruction error (mean and maximum) of differential training with varying number of hidden layers and fixed subspace training.

Layers	1	2	3	4	5
Differential	2.58×10^{-3}	1.54×10^{-3}	1.10×10^{-3}	1.03×10^{-3}	9.47×10^{-4}
Mean error	0.0240	0.0197	0.0189	0.0187	0.0182
Max error	0.700	0.633	0.667	0.664	0.541

We use an ablation study to evaluate the influence of the hidden layer numbers, varying the number of fully connected layers from 1 to 5 while fixing the subspace network and anchor points. The prediction and reconstruction error for character *Agent* for each of these is shown in Table 2. As shown, the prediction error decreases when the number of hidden layers increases, suggesting the improvement of network capacity for fitting. Also observable is the decrease of the reconstruction error, but it is less significant compared with the reduction of prediction error, suggesting that the accuracy of differential training is not the bottleneck for reconstruction.

4.2 Evaluation for Subspace Training

We use character *Agent* to evaluate how the anchor points and subspace network influence the deformation approximation, considering different number of anchor points, selection methods and subspace network structures. For experiment purpose, we fix the differential training (4403 mesh vertices with 220PC) and only change the subspace network. We specify the number of anchor points as 1%, 2% and 5% of the mesh vertex count, similar to our evaluation of PCA for differential training. We report both the prediction and reconstruction error in Table 3. Notice we increase the percentage by adding new anchor points into the existing ones instead of selecting a new group. To compare the network structure, we conduct the subspace training using a single network instead of the subspace mini-networks ("2%Single"). The single network takes the entire vectorized features as input and outputs the deformation of all anchor points together. To compare different anchor point selection methods, we use a new group of anchor points around the scalp with less significant deformation ("2% Scalp"). Notice the original group of anchor points are selected on the face to cover major facial features with large deformation, as discussed in Section 3.2.

As observed, increasing the number of anchor points leads to higher prediction error since the network performs better fitting when the dimension is low. However, the reconstruction error stays roughly the same when the number of anchor point gets larger, because increasing the number of anchor points can improve the Laplacian matrix condition for reconstruction, which balances the increase of prediction error. We use 2% anchor points as a middle point for our implementation and the rest of the evaluation.

Fig. 4. Training poses for production characters *Agent* (top), *Matador* (middle) and *Bull* (bottom). The poses are generated from a broad sampling of the rig parameter space. Although many look implausible, they are necessary to capture the full space accurately without assumptions on the artist's control range.

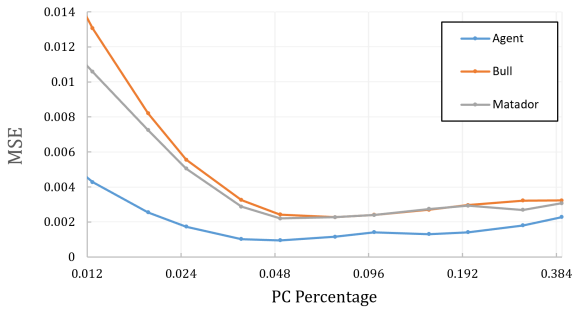

Fig. 5. Prediction error of differential network with varying PC percentage

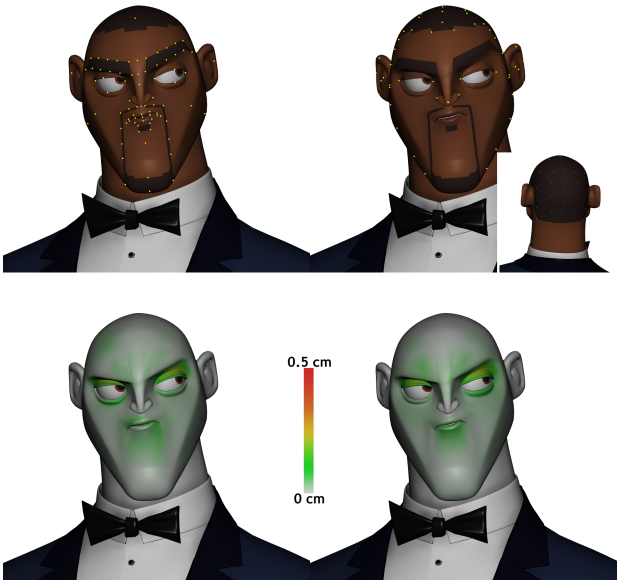

Fig. 6. Comparison of our method using anchors (in yellow) selected from the major facial features (left) vs. the less deformed scalp (right).

For the network comparison, both the prediction and reconstruction error of the subspace mini-networks ("2%") are lower than the single network ("2%Single"). We believe the dimension reduction is the reason for resulting performance improvement. The subspace mini-networks fit anchor points separately because they are disconnected and do not have direct spatial relationship, which enables better approximation. The single network, on the contrary, tries to learn the deformation of anchor points all at once, increasing the difficulty of fitting.

For different anchor point selection, we find using vertices with less deformation can cause larger reconstruction error even when the prediction error is smaller. The network has better performance because no deformation needs to be learned for those vertices, but they are not ideal for the reconstruction. Fig. 6 shows an example. As we can see, the deformation on mouth and eyelids are shifted when vertices on the scalp are selected as anchor points. Ideally, we want the anchor points to "nail" the deformed mesh in place and prevent large shifts or rotations for important face regions. Therefore, we select anchor points to cover major facial features with large deformation.

4.3 Results

In this section, we evaluate the accuracy of deformation reconstruction using well-animated poses from production. We evaluate the mean and max reconstruction errors over a series well-animated production sequences, where the deformations are much more exaggerated and dynamic. We present the quantities results in Table 5. The deformations of character *bull* are observed with larger errors because we test it on the most extreme animation sequence. Fig. 7 shows an example for the character and please refer to the supplemental video for detailed comparison. In general, our method can accurately reconstruct mesh surface with mean errors smaller

Table 3. Prediction error (subspace) and reconstruction error (mean and max) of the subspace training with varying anchor percentage with fixed differential training.

	1%	2%	2%(Single)	2%(Scalp)	5%
Subspace	1.35×10^{-3}	$\mathbf{1.71 \times 10^{-3}}$	7.42×10^{-3}	9.37×10^{-4}	1.73×10^{-3}
Mean error	0.0207	**0.0186**	0.0336	0.0192	0.0158
Max error	0.524	**0.517**	0.657	0.562	0.577

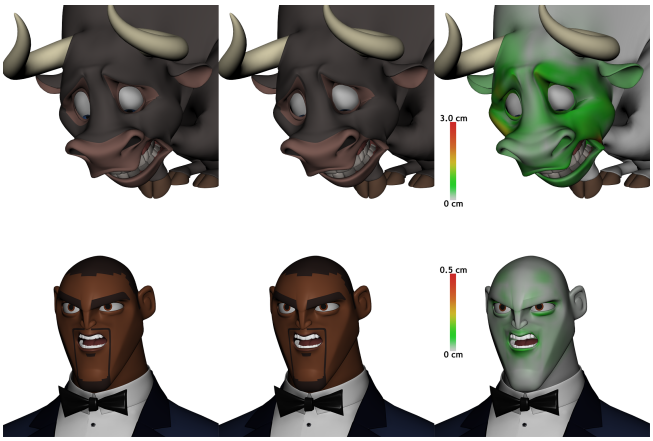

Fig. 7. Side-by-side comparison of ground truth (left), our approximation (center), and heatmap indicating per-vertex distance error in cm (right).

Table 4. Prediction errors (Differential and Subspace) and reconstruction errors (Mean and Max) for the tests with different training data size.

	25%	50%	75%	100%
Differential	2.66×10^{-3}	1.57×10^{-3}	1.03×10^{-3}	1.53×10^{-3}
Subspace	3.91×10^{-3}	2.81×10^{-3}	2.07×10^{-3}	1.71×10^{-3}
Mean error	0.0301	0.0246	0.0195	0.0186
Max error	0.891	0.819	0.740	0.517

Table 5. Mean and max reconstruction absolute errors evaluated on the well-animated production sequences, and as a percentage of face height.

	Agent	Bull	Matador
Mean error	0.032	0.512	0.087
Percentage	0.127%	0.607%	0.334%
Max error	0.630	4.682	0.782
Percentage	2.50%	5.55%	3.00%
Number of Poses	808	249	359

than 0.6% and max error smaller than 6% of the size of the character faces.

As a data-driven solution, the accuracy of our model largely relies on sufficient training data. To evaluate how the training size influence the performance, we alternatively reduce the size for character *Agent* to be 25%, 50% and 75% of the original dataset while keeping the test data unchanged (200 randomly generated poses). We present both the prediction errors for the differential and subspace training and reconstruction errors in Table 4. Indeed, the increasing of training data will boost the performance. However, the improvement is not very significant when increasing the size over 75%.

4.4 Comparison

We first compare the accuracy of facial deformation approximation with previous methods. Then we apply our method to body rigs and compare the results with Bailey et al. [2018].

4.4.1 Facial Deformation Comparison. We compare our method with linear blend skinning (LBS), PCA with linear regression (PCA), local Cartesian coordinates training using our model (Local) and Meyer et al. [2007] (KPSA). KPSA is an example-based deformation approximation method, which uses the deformation of key points as input to PCA to derive vertex positions for the entire mesh. The quality of the training data significantly influences the accuracy of the deformation, and their method relies on evaluating the original deformer stack to determine the key points on the fly. For the Local model, we apply the same differential network with PCA directly on the vertex local offsets without converting them into differential coordinates. No subspace learning and reconstruction is required for this model. We use it to compare the differential training and evaluate the contribution of mesh representation. We use the same set of randomly-generated training poses as used by our model to train both KPSA and the Local model, and we perform evaluation on the same well-animated sequences introduced in the last subsection.

Table 6. Mean and max reconstruction errors using our method compared with Linear Blend Skinning (LBS), PCA with linear regression, our model using local offset for training (Local) and Meyer et al. [2007] (KPSA). The comparison is shown for a set of test poses from a well-animated production sequence.

	Agent		Bull		Matador	
	Mean	Max	Mean	Max	Mean	Max
LBS	0.174	3.228	1.672	23.56	0.228	4.261
PCA	0.073	1.980	0.848	8.367	0.158	1.533
Local	0.072	0.689	0.521	5.779	0.155	1.106
KPSA	0.061	1.623	2.115	34.25	0.089	1.664
Ours	0.032	0.630	0.512	4.682	0.087	0.782

We report the reconstruction error in Table 6 and provide visual comparison in Fig. 8. As observed, our method outperforms the other four methods in both quantitative and visualized results. We use the result of LBS as a base-line as it does not provide any non-linear deformation. From the heat map, we can see that the Local model fails to capture the local deformation on the eyelids and the mouth is shifted. This is because no neighbor vertex information is embedded in the local offset, which makes it difficult for the network to predict the local deformation. For KPSA, it fails to reconstruct the deformation in the eyebrow region and the corner of the lips, even though with a substantial increase in the number of key points (274) and basis vectors (200) used in the original example. The relatively poor performance is caused by the linear reconstruction of training data, which could only provide a limited range and a fixed dimension for the approximated deformation. Once the target pose is out of the dimension defined by the PCA, it is difficult for that method to achieve high reconstruction accuracy. Additionally, the key points still need to be driven by the original rig. In comparison, our method can accurately capture the local deformation because of the error characteristics of the differential coordinates. Due to the nonlinear fitting capability of deep neural networks, our method can use randomly generated data for training and approximate deformation with a much larger range.

4.4.2 Body Deformation Comparison. We demonstrate our method applied to body deformation approximation and compare our results with Bailey et al. [2018]. We use character *Agent* as the example for comparison. The character's height is 200.25 cm. The body contains 4908 vertices and the rig includes 107 joints controls with hand

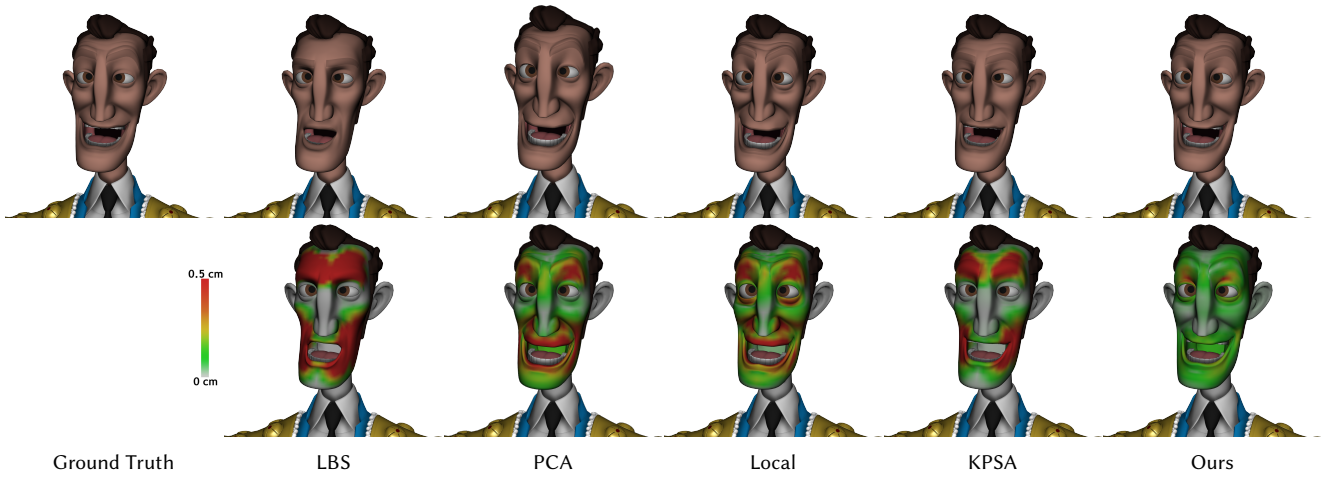

Fig. 8. Comparisons for facial deformation between ground truth, Linear Blend Skinning (LBS), PCA with linear regression, local offset training, KPSA, and our method using a well-animated pose from production.

joints excluded. We use the same training method and network structures mentioned in Section 3.2 and generate random poses for body rig as training data. Since the body rig does not include numerical controls, we remove them from input and only vectorize the joint controls. We use 245 PCs for the differential training and select 118 anchor points that are well-distributed around all the joints of the body. For Bailey et al. [2018], denoted as FDDA, we follow their methods to train multiple small networks (2 hidden layers with 128 units), each of which corresponds to a joint control and predicts the nonlinear deformation of the neighbor vertices in local coordinates. We generate 9800 random poses using the method described in Section 3.3 as training data and perform the evaluation using 189 poses from a well-animated production sequence for all three models.

We report the mean and max reconstruction error in the inline table and we show deformation results in Fig 9. The results indicate that our method outperforms the FDDA method, especially for the maximum error. Using multiple networks for deformation approximation, FDDA suffers from discontinuity problem on torso and left arm. We can observe high errors on the connecting parts of the body since the vertices from the two parts are predicted by different networks. The discontinuity is caused by the slight change of joint scales in the evaluation sequence, which does not show up in the training data. Due to the local joint input and small-scale network, FDDA suffers from overfitting to the training data and is sensitive to new values. Our method uses a deeper network with a much larger input size, which increases the capacity and makes the network less sensitive to the unseen scaling change of a couple joints. Since our method also uses small networks for subspace training, there might be some anchors that are affected by the scaling. But due to the least square reconstruction, the local error is nicely distributed as low frequency error and is much less noticeable.

	Mean	Max
Ours	0.217	4.17
FDDA	0.263	6.41

Increasing the network size for FDDA may improve the overall performance, however evaluating a large number of deeper networks (40 in our case) would cause significant performance downgrade.

Fig. 10 shows the error distribution of each model. As observed, the error distribution of our model is compressed to the lower range while the distribution of FDDA extends to large errors. Although the two methods have similar mean error, this observation suggests that our method can provide smooth approximation results with smaller maximum errors, and avoids inappropriate deformation.

5 CONCLUSION

In this paper we have presented a learning-based solution to capture facial deformation for rigs with high accuracy. Our method uses differential coordinates and a learned subspace to reconstruct smooth nonlinear facial deformation. We have demonstrated the robustness of our method on a wide range of animated poses. Our method compares favorably with existing solutions for both facial and body deformation. We also have successfully integrated our solution into the production pipeline.

Our work has limitations that we wish to investigate in the future. First, our method needs manually-selected anchor points for subspace training and reconstruction. It would be interesting to investigate methods for inferring the anchor points based on the characteristics of the facial mesh and training poses. Second, as a deep learning based approach, a model must to be trained for every character with different rig behavior or mesh topology. We would like to explore the possibility of integrating a high level super-rig into our method to provide a single model for different characters.

REFERENCES

Steven S An, Theodore Kim, and Doug L James. 2008. Optimizing cubature for efficient integration of subspace deformations. In *ACM transactions on graphics (TOG)*, Vol. 27. ACM, 165.

Stephen W Bailey, Dave Otte, Paul Dilorenzo, and James F O'Brien. 2018. Fast and deep deformation approximations. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 119.

Jernej Barbič and Doug L James. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM transactions on graphics (TOG)* 24, 3 (2005), 982–990.

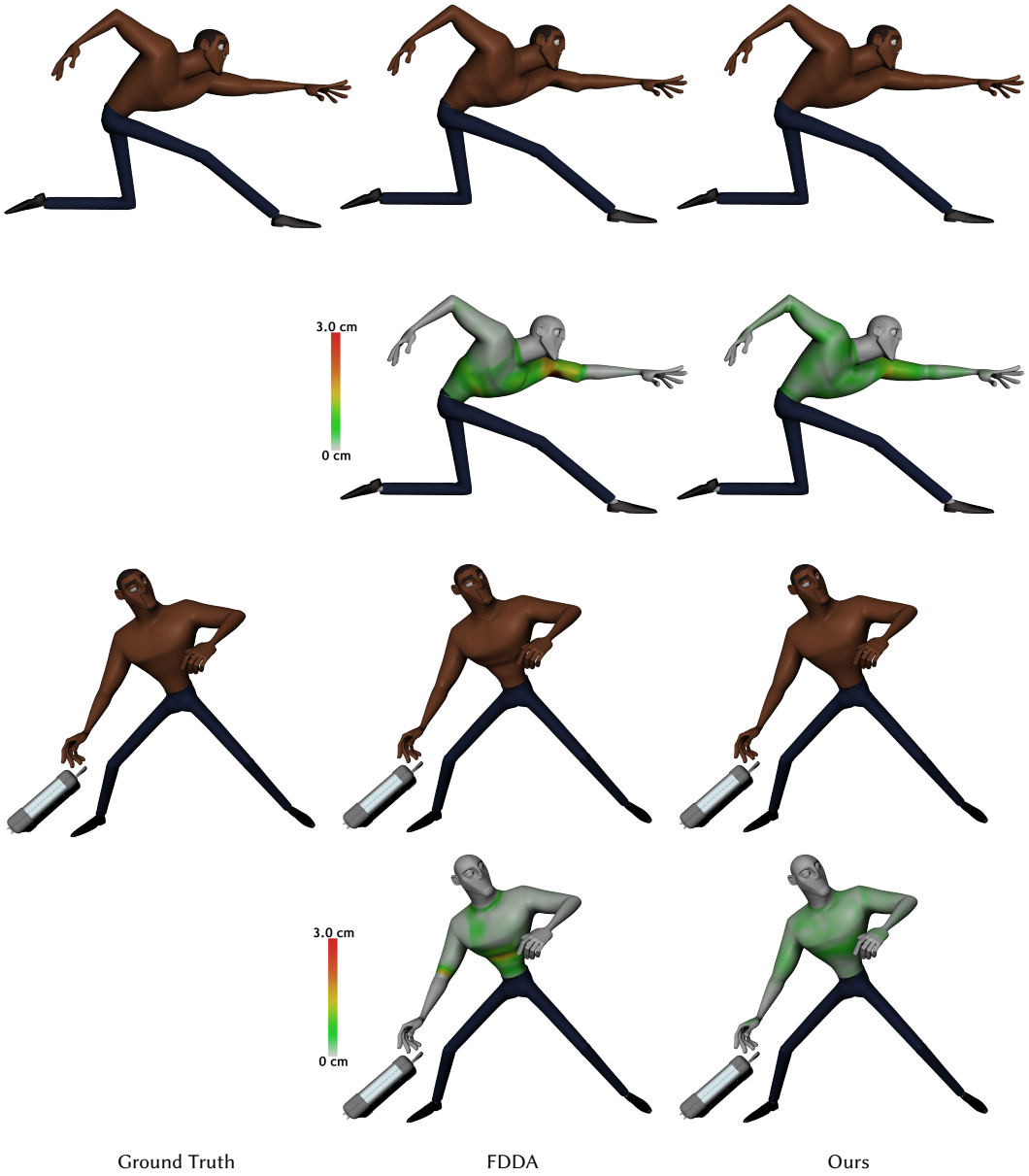

Fig. 9. Comparisons for body deformation between the ground truth, Bailey et al. [2018] (FDDA) and our method using well-animated poses.

Jernej Barbič, Funshing Sin, and Eitan Grinspun. 2012. Interactive editing of deformable simulations. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 70.

Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-reduced projective dynamics. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 80.

Doron Chen, Daniel Cohen-Or, Olga Sorkine, and Sivan Toledo. 2005. Algebraic analysis of high-pass quantization. *ACM Transactions on Graphics (TOG)* 24, 4 (2005), 1259–1282.

Matthew Cong, Kiran S Bhat, and Ronald Fedkiw. 2016. Art-directed muscle simulation for high-end facial animation. In *Symposium on Computer Animation*. 119–127.

Zhigang Deng, Pei-Ying Chiang, Pamela Fox, and Ulrich Neumann. 2006. Animating blendshape faces by cross-mapping motion capture data. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*. ACM, 43–48.

Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. 2018. Automatic unpaired shape deformation transfer. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 237.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.* 30, 4 (2011), 78.

Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. 2007. Harmonic coordinates for character articulation. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 71.

Pushkar Joshi, Wen C Tien, Mathieu Desbrun, and Frédéric Pighin. 2006. Learning controls for blend shape based realistic facial animation. In *ACM Siggraph 2006 Courses*. ACM, 17.

Tao Ju, Scott Schaefer, and Joe Warren. 2005. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 561–566.

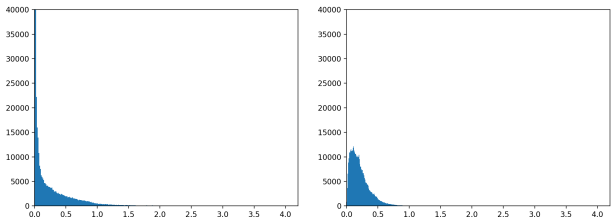

Fig. 10. Comparison of error distribution for body deformation using well-animated pose from production.

Ladislav Kavan, Steven Collins, and Carol O'Sullivan. 2009. Automatic linearization of nonlinear skinning. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*. ACM, 49–56.

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2008. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)* 27, 4 (2008), 105.

Ladislav Kavan and Olga Sorkine. 2012. Elasticity-inspired deformers for character articulation. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 196.

Ladislav Kavan and Jiří Žára. 2005. Spherical blend skinning: a real-time deformation of articulated models. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*. ACM, 9–16.

Meekyoung Kim, Gerard Pons-Moll, Sergi Pujades, Seungbae Bang, Jinwook Kim, Michael J Black, and Sung-Hee Lee. 2017. Data-driven physics for human soft tissue animation. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 54.

Petr Krysl, Sanjay Lall, and Jerrold E Marsden. 2001. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *International Journal for numerical methods in engineering* 51, 4 (2001), 479–504.

Samuli Laine, Tero Karras, Timo Aila, Antti Herva, Shunsuke Saito, Ronald Yu, Hao Li, and Jaakko Lehtinen. 2017. Production-level facial performance capture using deep convolutional neural networks. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 10.

Manfred Lau, Jinxiang Chai, Ying-Qing Xu, and Heung-Yeung Shum. 2009. Face poser: Interactive modeling of 3d facial expressions using facial priors. *ACM Transactions on Graphics (TOG)* 29, 1 (2009), 3.

Binh Huy Le and Jessica K Hodgins. 2016. Real-time skeletal skinning with optimized centers of rotation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 37.

Binh Huy Le and JP Lewis. 2019. Direct delta mush skinning and variants. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 113.

John P Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Frederic H Pighin, and Zhigang Deng. 2014. Practice and Theory of Blendshape Facial Models. *Eurographics (State of the Art Reports)* 1, 8 (2014), 2.

John P Lewis and Ken-ichi Anjyo. 2010. Direct manipulation blendshapes. *IEEE Computer Graphics and Applications* 30, 4 (2010), 42–50.

John P Lewis, Matt Cordner, and Nickson Fong. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 165–172.

Hao Li, Thibaut Weise, and Mark Pauly. 2010. Example-based facial rigging. In *Acm transactions on graphics (tog)*, Vol. 29. ACM, 32.

Yaron Lipman, David Levin, and Daniel Cohen-Or. 2008. Green coordinates. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 78.

Lijuan Liu, Youyi Zheng, Di Tang, Yi Yuan, Changjie Fan, and Kun Zhou. 2019. NeuroSkinning: automatic skin binding for production characters with deep graph networks. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 114.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2015. SMPL: A skinned multi-person linear model. *ACM transactions on graphics (TOG)* 34, 6 (2015), 248.

Ran Luo, Tianjia Shao, Huamin Wang, Weiwei Xu, Xiang Chen, Kun Zhou, and Yin Yang. 2018. NNWarp: Neural Network-based Nonlinear Deformation. *IEEE transactions on visualization and computer graphics* (2018).

Nadia Magnenat-Thalmann, Richard Laperrire, and Daniel Thalmann. 1988. Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface'88*. Citeseer.

Joe Mancewicz, Matt L Derksen, Hans Rijpkema, and Cyrus A Wilson. 2014. Delta Mush: smoothing deformations while preserving detail. In *Proceedings of the Fourth Symposium on Digital Production*. ACM, 7–11.

Bruce Merry, Patrick Marais, and James Gain. 2006. Animation space: A truly linear framework for character animation. *ACM Transactions on Graphics (TOG)* 25, 4 (2006), 1400–1423.

Mark Meyer and John Anderson. 2007. Key point subspace acceleration and soft caching. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 74.

Tomohiko Mukai. 2015. Building helper bone rigs from examples. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*. ACM, 77–84.

Tomohiko Mukai and Shigeru Kuriyama. 2016. Efficient dynamic skinning with low-rank helper bone controllers. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 36.

Alexander Pentland and John Williams. 1989. Good vibrations: Modal dynamics for graphics and animation. (1989).

Weiguang Si, Sung-Hee Lee, Eftychios Sifakis, and Demetri Terzopoulos. 2014. Realistic biomechanical simulation and control of human swimming. *ACM Transactions on Graphics (TOG)* 34, 1 (2014), 10.

Peter-Pike J Sloan, Charles F Rose III, and Michael F Cohen. 2001. Shape by example. In *Proceedings of the 2001 symposium on Interactive 3D graphics*. ACM, 135–143.

Olga Sorkine. 2005. Laplacian mesh processing. In *Eurographics (STARs)*. 53–70.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, Vol. 4. 109–116.

Olga Sorkine, Daniel Cohen-Or, Dror Irony, and Sivan Toledo. 2005. Geometry-aware bases for shape approximation. *IEEE transactions on visualization and computer graphics* 11, 2 (2005), 171–180.

Robert W Sumner, Johannes Schmid, and Mark Pauly. 2007. Embedded deformation for shape manipulation. In *ACM SIGGRAPH 2007 papers*. 80–es.

Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. 2018a. Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5841–5850.

Qingyang Tan, Lin Gao, Yu-Kun Lai, Jie Yang, and Shihong Xia. 2018b. Mesh-based autoencoders for localized deformation component analysis. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Robert Y Wang, Kari Pulli, and Jovan Popović. 2007. Real-time enveloping with rotational regression. In *ACM Transactions on Graphics (TOG)*, Vol. 26. ACM, 73.

Xiaohuan Corina Wang and Cary Phillips. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, 129–138.

Yu Wang, Alec Jacobson, Jernej Barbič, and Ladislav Kavan. 2015. Linear subspace design for real-time shape deformation. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.

Thibaut Weise, Sofien Bouaziz, Hao Li, and Mark Pauly. 2011. Realtime performance-based facial animation. In *ACM transactions on graphics (TOG)*, Vol. 30. ACM, 77.

Hao Zhang, Oliver Van Kaick, and Ramsay Dyer. 2010. Spectral mesh processing. In *Computer graphics forum*, Vol. 29. Wiley Online Library, 1865–1894.