A Majorization-Minimization Based Method for Nonconvex Inverse Rig Problems in Facial Animation: Algorithm Derivation

Stevo Racković[1], Claudia Soares[2], Dušan Jakovetić[3], Zoranka Desnica[4]
[1]Institute for Systems and Robotics, Instituto Superior Técnico, Lisbon, Portugal
[2]Computer Science Department, NOVA School of Science and Technology, Caparica, Portugal
[3]Department of Mathematics, Faculty of Sciences, University of Novi Sad, Novi Sad, Serbia
[4]3Lateral Animation Studio, Epic Games Company
e-mail: `stevo.rackovic@tecnico.ulisboa.pt`

Abstract – **Automated methods for facial animation are a necessary tool in the modern industry since the standard blendshape head models consist of hundreds of controllers and a manual approach is painfully slow. Different solutions have been proposed to produce an output in real-time or generalize well for different face topologies. However, all these prior works consider a linear approximation of the blendshape function and hence do not provide a high-enough level of details for modern realistic human face reconstruction. We build a method for solving the inverse rig in blendshape animation using the quadratic corrective terms, which increases the accuracy. At the same time, due to the proposed construction of the objective function, it yields the sparser estimated weight vector compared to the state-of-the-art methods. The former feature means lower demand for subsequent manual corrections of the solution, while the latter indicates that the manual modifications are also easier to include. Our algorithm is iterative, based on an increment optimization and employs a Majorization Minimization paradigm to cope with the increased complexity produced by adding the corrective terms. The surrogate function is easy to solve and allows for further parallelization on the component level within each iteration. This paper is complementary to an accompanying paper [20] where we provide detailed experimental results and discussion, including highly-realistic animation data and show a clear superiority of the results compared to the state-of-the-art methods.**

Keywords – **Inverse Rig, Quadratic Blendshape Model, Majorization-Minimization**

I. Introduction

A human face animation is an increasingly popular field of research within the applied mathematics community, of interest not only for the production of movies and video games but also in virtual reality, education, communication, etc. A common approach to face animation is using blendshapes [4], [15] since such models provide intuitive controls, even though building the model is demanding in terms of time and effort [17], [18], [28], [30]. Inverse rig estimation is one of the aspects in blendshape animation that can be automated hence it is often addressed in the literature. Possible approaches for solving the inverse rig can be classified into data-based and model-based techniques, where the first group demands large amounts of animation data for training purposes, and the second group works with only a blendshape function and the basis vectors. While various machine learning techniques show good performance [1], [3], [7], [9]–[11], [13], [21], [24], [26], [29], such methods demand long animated sequences that cover all the regular facial expressions, to train a good regressor. This often makes them infeasible or unprofitable. Conversely, model-based approaches [4], [6], [16], [23] rely on applying optimization techniques and do not demand training data. Yet, a precise rig function or a good approximation is necessary to provide high-quality results. Without exception, all the papers that propose model-based solutions work with linear blendshape function, which does not offer high-enough fidelity for realistic animated faces.

We proposed a new model-based method for solving the inverse rig problem such that it includes the quadratic corrective terms, which leads to higher accuracy of the fit compared to the standard linear rig approximation [11], [25]. Method utilizes the common optimization tools, namely increment optimization and Majorization-Minimization [14], [31].

A. Contributions

In the companion paper [20], we present a novel method for solving the inverse rig problem when the blendshape model is assumed to be quadratic. This method targets a specific subdomain of facial animation — highly-realistic human face models used in movie and video games production. Here the accurate fit plays a more critical role than the execution time. For this reason, the added complexity of a quadratic blendshape rig is justified since it significantly increases the mesh fidelity of the result. Besides increasing the mesh accuracy, our solution yields fewer activated components than the state-of-the-art methods, which is another desirable property in production. The main contributions of the current paper are to provide a detailed derivation of the proposed inverse rig method and describe results on its convergence guarantees. We refer to [20] for further practical and implementation aspects, as well as extensive numerical results on real-world animation data.

The rest of the paper is organized as follows. Section II. formulates the inverse rig problem and covers the existing

This work has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 812912, and from strategic project NOVA LINCS (FCT UIDB/04516/2020). The work has also been supported in part by the Ministry of Education, Science and Technological Development of the Republic of Serbia (Grant No. 451-03-9/2021-14/200125).

solutions from the literature. Section III introduces our algorithm and gives a detailed derivation of each step. Finally, we conclude the paper in Section IV.

II. Problem formulation and preliminaries

The main components of the blendshape model are the *neutral mesh* $\mathbf{b}_0 \in \mathbb{R}^{3n}$ sculpted by an artist, as well as a set of m blendshapes $\mathbf{b}_1, ..., \mathbf{b}_m \in \mathbb{R}^{3n}$, where n is the number of vertices in the mesh. Blendshapes are topologically identical copies of a neutral mesh but with some vertices displaced in space to simulate a local deformation of the face. The offset between neutral mesh and blendshapes yields delta blendshapes $\Delta\mathbf{b}_i = \mathbf{b}_0 + \mathbf{b}_i$, for $i = 1, ..., m$, that are added on top of a neutral mesh $\mathbf{b}_0$, with a weight $w_i \in [0,1]$, to produce an effect of a local deformation as $\mathbf{b}_0 + w_i\Delta\mathbf{b}_i$. Multiple local deformers are usually combined to produce more complex facial expressions. A blendshape function then can be defined as

$$f_L(\mathbf{w}) = \mathbf{b}_0 + \sum_{i=1}^{m} w_i \Delta\mathbf{b}_i = \mathbf{b}_0 + \mathbf{B}\mathbf{w}, \tag{1}$$

where $\mathbf{w} = [w_1, ..., w_m]^T$ is a weight vector and $\mathbf{B} = [\Delta\mathbf{b}_1, ..., \Delta\mathbf{b}_m]$ is a blendshape matrix. The subscript L indicates that this is a linear model, while for realistic face representation it is common to consider a more complex form with quadratic terms, as explained in the following lines.

Some pairs of blendshapes, $\mathbf{b}_i$ and $\mathbf{b}_j$, with overlapping region of influence, might produce artifacts on the face (mesh breaking or giving an unbelievable deformation), hence a corrective term $\mathbf{b}^{\{i,j\}}$ needs to be included to fix any issues. Now, whenever the two blendshapes are activated simultaneously, the corrective term is added as well, with a weight equal to the product of two corresponding weights. A quadratic blendshape function is then defined as:

$$f_Q(\mathbf{w}) = b_0 + \sum_{i=1}^{m} w_i \Delta\mathbf{b}_i + \sum_{(i,j)\in\mathscr{P}} w_i w_j \mathbf{b}^{\{i,j\}} \tag{2}$$

where $\mathscr{P}$ is a set of indices of the blendshapes that invoke a corrective term.

In production, it might be essential to solve the inverse problem. That is, considering there is a given mesh $\widehat{\mathbf{b}}$, that is conventionally obtained either as a 3D scan of an actor or a capture of the face markers, one needs to estimate a configuration of the weight vector $\mathbf{w}$ that produces a mesh as similar as possible to $\widehat{\mathbf{b}}$. This problem is known as the inverse rig problem or the automatic keyframe animation. As we will discuss in the next section, it is common to pose this in a least-squares setting.

A. Existing solutions

When we consider a model-based solution to the inverse rig problem, the state-of-the-art method is [5], where the optimization problem is formulated as regularized least-squares minimization:

$$\underset{\mathbf{w}}{\text{minimize}}\, \|\mathbf{B}\mathbf{w} - \widehat{\mathbf{b}}\|^2 + \alpha\|\mathbf{w}\|^2. \tag{3}$$

Regularization is necessary because otherwise the solution is not unique or at least stable. Additionally, it is desired to keep the number of non-zero elements of $\mathbf{w}$ lower because it allows for further manual editing, which is common in animation. The solution to (3) is given in a closed-form as

$$\mathbf{w} = (\mathbf{B}^T\mathbf{B} + \alpha\mathbf{I})^{-1}\mathbf{B}^T\widehat{\mathbf{b}}. \tag{4}$$

One adjustment to this solution is given in the same paper. Authors approximate a blendshape matrix $\mathbf{B}$ with a sparse matrix $\mathbf{B}^{loc}$. It is obtained by applying a heat kernel over the rows of an original blendshape matrix. This sets low values to zero, meaning that effects of the least significant blendshapes are excluded for each vertex.

A different approach is given in [22], where components are visited and optimized sequentially, and after each iteration $i = 1, ..., m$, a residual term is updated:

$$\begin{aligned} &\text{step 1:}\ \underset{w_i}{\text{minimize}}\, \|w_i\Delta\mathbf{b}_i - \widehat{\mathbf{b}}\|^2, \\ &\text{step 2:}\ \widehat{\mathbf{b}} \leftarrow \widehat{\mathbf{b}} - w_i\Delta\mathbf{b}_i. \end{aligned} \tag{5}$$

Here *step 1* finds the optimal activation of a single controller w_i, and *step 2* removes its effect for subsequent iterations. This yields a sparse weight vector and excludes the possibility of simultaneously activating mutually exclusive blendshapes (like `mouth-corner-up` and `mouth-corner-down`). However, the order in which the blendshapes are visited is crucial to obtain an acceptable solution. The authors propose ordering them based on the average offset that each blendshape causes over the whole face.

Other papers consider approaches similar to (3) or (5), and they all assume a linear blendshape function. A linear model has an advantage over a quadratic because it gives rise to a convex problem, and it is thus simple and easy to work with; however, that simplicity comes at a price — it does not provide enough detail for realistic human face representation in high-quality movies and video games. In the next section, we introduce our solution of the inverse rig that takes into account quadratic corrective terms.

III. Proposed solution

This section presents a detailed derivation of our method that utilizes second-order blendshape models; we refer to [20] to detailed method's presentation from the domain point of view and for extensive numerical experiments on the method. Our algorithm targets specifically a high-quality realistic human face animation, hence we assume that a real-time execution is not a priority and that the activation weights are strictly bound to $[0,1]$ interval. We first explain the algorithm derivation and then detail each algorithm step. The optimization problem looks for the optimal weigh vector configuration $\mathbf{w}$ that fits on a given mesh $\widehat{\mathbf{b}}$, assuming a

Many authors neglect this constraint with a justification that the values outside this interval can be still used for *exaggerated* expressions in animated characters. In our setting, this is not allowed by the construction of the models and we have to take these constraints into account.

quadratic blendshape function (2), as

$$\underset{\mathbf{0}\leq\mathbf{w}\leq\mathbf{1}}{\text{minimize}}\,\|f_Q(\mathbf{w})-\widehat{\mathbf{b}}\|^2+\alpha\mathbf{1}^T\mathbf{w}, \tag{6}$$

where the first term is a data fidelity and the second is regularization. The regularization term with $\alpha > 0$ enforces a low cardinality of the solution vector. The problem is approached in a manner of increment optimization, i.e., it is an iterative process where we choose the initial vector of controller weights $\mathbf{w}_{(0)} \in \mathbb{R}^m$, and at each iteration $t = 0,...,T$, solve for an optimal increment vector $\mathbf{v} \in \mathbb{R}^m$ such that the increment $\mathbf{v}$ solves the following optimization problem:

$$\underset{-\mathbf{w}_{(t)}\leq\mathbf{v}\leq\mathbf{1}-\mathbf{w}_{(t)}}{\text{minimize}}\;\|f_Q(\mathbf{w}_{(t)}+\mathbf{v})-\widehat{\mathbf{b}}\|^2+\alpha\mathbf{1}^T(\mathbf{w}_{(t)}+\mathbf{v}). \tag{7}$$

The weights vector is updated as $\mathbf{w}_{(t+1)} = \mathbf{w}_{(t)} + \mathbf{v}$, and the process is repeated until convergence. Under the quadratic approximation of the rig function, the objective function in (7) is fairly complex, hence we simplify it by applying majorization minimization (MM). That is, to solve (7), we define an upper bound function $\psi(\mathbf{v};\mathbf{w}) : \mathbb{R}^m \to \mathbb{R}$ over the objective function in (7) such that it is easier to minimize and satisfies conditions 1 and 2 given below. Note that these conditions define a class of functions $\psi(\mathbf{v};\mathbf{w})$ as potential majorizers to the objective (7), but later in this section, we define a specific choice of $\psi(\cdot)$ used in the proposed algorithm.

Condition 1. For any feasible vector $\mathbf{0} \leq \mathbf{w} \leq \mathbf{1}$, for all the values of an increment vector $\mathbf{v}$ such that $\mathbf{0} \leq \mathbf{w}+\mathbf{v} \leq \mathbf{1}$, the following holds:

$$\|f_Q(\mathbf{w}+\mathbf{v})-\widehat{\mathbf{b}}\|^2+\alpha\mathbf{1}^T(\mathbf{w}+\mathbf{v}) \leq \psi(\mathbf{v};\mathbf{w}). \tag{8}$$

Condition 2. The upper bound $\psi(\mathbf{v};\mathbf{w})$ matches the value of the objective (7) at point $\mathbf{v} = \mathbf{0}$, i.e.,

$$\|f_Q(\mathbf{w})-\widehat{\mathbf{b}}\|^2+\alpha\mathbf{1}^T(\mathbf{w}) = \psi(\mathbf{0};\mathbf{w}). \tag{9}$$

In the rest of the paper we will write $\psi(\mathbf{v})$ instead of $\psi(\mathbf{v};\mathbf{w})$, for the sake of simplicity. Further we proceed with the problem in the form

$$\underset{-\mathbf{w}\leq\mathbf{v}\leq\mathbf{1}-\mathbf{w}}{\text{minimize}}\,\psi(\mathbf{v}), \tag{10}$$

and the following proposition gives us guarantees that such an approach leads to the minimization of the original objective.

Proposition 1. Under Conditions 1 and 2, a sequence of iterates $\mathbf{w}_{(t)}$ for $t \in \mathbb{N}$ obtained as the solutions to problem (10) (with $\mathbf{w}_{(t+1)} = \mathbf{w}_{(t)} + \mathbf{v}_{(t)}$) produces a nonincreasing sequence of values of the objective (6), i.e.,

$$\|f_Q(\mathbf{w}_{(t)})-\widehat{\mathbf{b}}\|^2+\alpha\mathbf{1}^T\mathbf{w}_{(t)} \geq \|f_Q(\mathbf{w}_{(t+1)})-\widehat{\mathbf{b}}\|^2+\alpha\mathbf{1}^T\mathbf{w}_{(t+1)}, \tag{11}$$

for $t \in \mathbb{N}$.

Upper Bound Function. Let us now introduce a specific majorizer that we apply in the proposed algorithm, and below we will give a complete derivation. We define the upper bound function of the objective (7) as

$$\psi(\mathbf{v}) = \sum_{i=1}^{3n}\psi_i(\mathbf{v})+\alpha\mathbf{1}^T(\mathbf{w}+\mathbf{v}), \tag{12}$$

which is an original regularization term added on a sum of coordinate-wise upper bounds $\psi_i(\mathbf{v}) : \mathbb{R}^m \to \mathbb{R}$ of the data fidelity term, where n is a number of vertices in the mesh. The component-wise bounds have the form

$$\begin{aligned}\psi_i(\mathbf{v}) := g_i^2+2g_i\sum_{j=1}^{m}h_{ij}v_j+\\ 2\big(g_i\lambda_M(\mathbf{D}^{(i)},g_i)+\|\mathbf{h}_i\|^2\big)\sum_{j=1}^{m}v_j^2+2m\sigma^2(\mathbf{D}^{(i)})\sum_{j=1}^{m}v_j^4\end{aligned} \tag{13}$$

where $g_i := \mathbf{B}_i\mathbf{w}+\mathbf{w}^T\mathbf{D}^{(i)}\mathbf{w}-\widehat{b}_i$, and $\mathbf{h}_i := \mathbf{B}_i+2\mathbf{w}^T\mathbf{D}^{(i)}$ are introduced to simplify the notation; $\mathbf{D}^{(i)} \in \mathbb{R}^{m\times m}$ is a symmetric (and sparse) matrix whose nonzero entries are extracted from the corrective blendshapes as $D_{jk}^{(i)} = D_{kj}^{(i)} = \frac{1}{2}b_i^{\{j,k\}}$; the largest singular value of a matrix $\mathbf{D}^{(i)}$ is denoted $\sigma(\mathbf{D}^{(i)})$; a function $\lambda_M(\mathbf{D}^{(i)},g_i) : (\mathbb{R}^{m\times m},\mathbb{R}) \to \mathbb{R}$ is defined as

$$\lambda_M(\mathbf{D}^{(i)},g_i) := \begin{cases}\lambda_{\min}(\mathbf{D}^{(i)}) & \text{if } g_i < 0,\\ \lambda_{\max}(\mathbf{D}^{(i)}) & \text{if } g_i \geq 0,\end{cases}$$

where $\lambda_{\min}(\mathbf{D}^{(i)})$ represents the smallest and $\lambda_{\max}(\mathbf{D}^{(i)})$ the largest eigenvalue of $\mathbf{D}^{(i)}$. Under the surrogate function (12), the problem (10) can be analytically solved component-wise, where for each component $j = 1,...,m$, the objective is a scalar quartic equation:

$$\begin{aligned}&\underset{v_j}{\text{minimize}}\; qv_j+rv_j^2+sv_j^4,\\ &\text{s.t. } 0 \leq w_j+v_j \leq 1.\end{aligned} \tag{14}$$

The expressions for the polynomial coefficients q,r,s are

$$\begin{aligned}r &= 2\sum_{i=1}^{3n}\left(g_i\lambda_M(\mathbf{D}^{(i)},g_i)+\|\mathbf{h}_i\|^2\right),\\ s &= 2m\sum_{i=1}^{3n}\sigma_{\max}^2(\mathbf{D}^{(i)}),\\ q &= 2\sum_{i=1}^{3n}g_ih_{ij}+\alpha.\end{aligned} \tag{15}$$

Notice that the coefficient q depends on a coordinate j, so it has to be computed for each controller separately, while r and s are calculated only once per iteration. We can find the extreme values of the polynomial using the roots of the cubic derivative $q+2rv_j+4sv_j^3 = 0$, and check if they are within the feasible interval $[0,1]$, and compare with the polynomial values at the borders to get the constrained minimizer. The idea of component-wise optimization of the weight vector makes our approach somewhat similar to that of [22], yet we do not update the vector until all the components are estimated. This solves the issue of the order in which the controllers are visited and additionally gives a possibility for a parallel implementation of the procedure. Another difference is that [22] considers only a single run over the coordinates, while for us, this is only a single step — we refer to it as an *inner iteration*. In this sense, we start with an arbitrary and feasible vector and repeat the inner iteration multiple times to provide an increasingly good estimate of the solution to (7), in a manner similar to the trust-region method [19], [27]. A complete inner iteration is summarized in Algorithm 1. In the following lines, we will detail a complete derivation of the upper bound given in (12).

Derivation of the Upper Bound. If we write down the data fidelity term from (7) as a sum, and consider each element $i = 1,...,3n$ of the sum separately, we can represent it in a canonical quadratic form:

$$\sum_{i=1}^{3n}\left(\mathbf{B}_i\mathbf{w} + \sum_{(j,k)\in\mathscr{P}} w_j w_k b_i^{\{j,k\}} - \widehat{b}_i\right)^2 = \sum_{i=1}^{3n}\left(\mathbf{B}_i\mathbf{w} + \mathbf{w}^T\mathbf{D}^{(i)}\mathbf{w} - \widehat{b}_i\right)^2. \tag{16}$$

Define a function $\phi_i(\mathbf{w}) : \mathbb{R}^m \to \mathbb{R}$ as $\phi_i(\mathbf{w}) := (\mathbf{B}_i\mathbf{w} + \mathbf{w}^T\mathbf{D}^{(i)}\mathbf{w} - \widehat{b}_i)^2$, which is a single coordinate of the data fidelity term. When we add the increment vector $\mathbf{v}$ on top of the current weight vector $\mathbf{w}$, this yields:

$$\begin{aligned}\phi_i(\mathbf{w}+\mathbf{v}) &= \left(g_i + \mathbf{h}_i\mathbf{v} + \mathbf{v}^T\mathbf{D}^{(i)}\mathbf{v}\right)^2 \\ &= g_i^2 + 2g_i\mathbf{h}_i\mathbf{v} + 2g_i\mathbf{v}^T\mathbf{D}^{(i)}\mathbf{v} + \left(\mathbf{h}_i\mathbf{v} + \mathbf{v}^T\mathbf{D}^{(i)}\mathbf{v}\right)^2.\end{aligned} \tag{17}$$

The data fidelity term is a sum of functions $\phi_i(\mathbf{w})$, hence in order to bound the objective, we bound each element of the sum $\phi_i(\mathbf{w}+\mathbf{v}) \le \psi_i(\mathbf{v})$. Let us first consider two nonlinear terms of $\phi_i(\mathbf{w})$, and bound each of them separately. A bound over the quadratic term $2g_i\mathbf{v}^T\mathbf{D}^{(i)}\mathbf{v}$ depends on the sign of g_i:

$$2g_i\mathbf{v}^T\mathbf{D}^{(i)}\mathbf{v} \le 2g_i\lambda_M\left(\mathbf{D}^{(i)}, g_i\right)\|\mathbf{v}\|^2. \tag{18}$$

The bound over a quartic term $(\mathbf{h}_i\mathbf{v} + \mathbf{v}^T\mathbf{D}^{(i)}\mathbf{v})^2$ is obtained by repeatedly applying the Cauchy-Schwartz inequality three times:

$$\begin{aligned}(\mathbf{h}_i\mathbf{v} + \mathbf{v}^T\mathbf{D}^{(i)}\mathbf{v})^2 &\le 2(\mathbf{h}_i\mathbf{v})^2 + 2(\mathbf{v}^T\mathbf{D}^{(i)}\mathbf{v})^2 \\ &\le 2\|\mathbf{h}_i\|^2\|\mathbf{v}\|^2 + 2\|\mathbf{v}\|^4\|\mathbf{D}^{(i)}\|^2 \\ &\le 2\|\mathbf{h}_i\|^2\|\mathbf{v}\|^2 + 2m\sigma^2(\mathbf{D}^{(i)})\sum_{j=1}^{m} v_j^4.\end{aligned} \tag{19}$$

The component-wise bound $\psi_i(\mathbf{v})$ is then given by (13) and a complete upper bound is obtained by summing the component-wise bounds and adding regularization term (Eq. 12).

Feasibility. Let us now state another proposition showing that the above derived upper bound function is feasible for *Proposition 1*.

Algorithm 1 Inner Iteration

```
Require: Blendshape matrix $\mathbf{B} \in \mathbb{R}^{3n\times m}$, corrective blendshape matrices $\mathbf{D}^{(i)} \in \mathbb{R}^{m\times m}$ for $i = 1,...,3n$, target mesh $\widehat{\mathbf{b}} \in \mathbb{R}^{3n}$, regularization parameter $\alpha \ge 0$ and weight vector $\mathbf{w} \in [0,1]^m$.
Ensure: $\hat{\mathbf{v}}$ - an optimal increment vector as a solution to (14).
  Compute coefficients $q,r,s$ from Eq. () and solve for an optimal increment vector $\hat{\mathbf{v}}$:
  $r = 2\sum_{i=1}^{3n}(g_i\lambda_M(\mathbf{D}^{(i)}, g_i) + \|\mathbf{h}_i\|^2)$,
  $s = 2m\sum_{i=1}^{3n}\sigma^2(\mathbf{D}^{(i)})$,
  for $k = 1,...,m$ do
      $q = 2\sum_{i=1}^{3n} g_i h_{ij} + \alpha$
      $\hat{v}_k = \text{argmin}_v\, qv + rv^2 + sv^4$
          s.t. $-w_k \le v \le 1 - w_k$
  end for
  return $\hat{\mathbf{v}}$
```

Proposition 2. The surrogate function $\psi(\mathbf{v}) : \mathbb{R}^m \to \mathbb{R}$, defined as in (12), satisfies *Conditions 1* and *2*, and hence it satisfies *Propositon 1.*

Proof.

In (17)-(19) function $\psi(\cdot)$ is derived so that it bounds the objective function (16) from above, hence it satisfies *Condition 1* by construction.

Now recall that the data fidelity term is a sum of functions $\phi_i(\cdot)$. To prove that *Condition 2* is satisfied it suffices to show that $\phi_i(\mathbf{w}+\mathbf{0}) = \psi_i(\mathbf{0})$. From (17) we see that $\phi_i(\mathbf{w}+\mathbf{0}) = g_i^2$, and from (13) we get $\psi_i(\mathbf{0}) = g_i^2$, which proves it. Hence, a derived bound satisfies *Propositon 1.* ■

Complete Algorithm. As mentioned above, our solution is iterative, and based on applying Algorithm 1 until convergence. In each iteration t, the weight vector is updated by adding an estimated increment vector: $\mathbf{w}_{(t+1)} = \mathbf{w}_{(t)} + \mathbf{v}$. The algorithm terminates when either a given maximum number of iterations is reached, or the difference between the cost function in two consecutive iterations is below a given threshold value $\varepsilon > 0$. While any feasible vector $\mathbf{0} \le \mathbf{w} \le \mathbf{1}$ can be used for initialization, we can rely on domain knowledge to choose a good starting point leading to faster convergence, and these strategies are discussed in the companion paper [20]. A complete method is summarized in Algorithm 2. Notice that in one of the steps of the algorithm, we compute eigen and singular values of matrices $\mathbf{D}^{(i)}$. This computation is needed only once per animated character, and we can reuse the calculated values for each following frame that is to be fitted.

Corollary 1. Due to the constraints in (14), the estimate sequence $\mathbf{w}_{(t)}$ produced by Algorithm 2 is feasible to problem (6) at all iterations $t \in \mathbb{N}$, as long as the initial weight vector $\mathbf{w}_{(0)}$ is feasible.

As demonstrated numerically in [20], Algorithm 2 produces accurate and sparse solutions of the inverse rig problem.

IV. Conclusion

This paper introduced the first model-based method for solving the inverse rig problem using the quadratic blendshape function. The proposed algorithm is based on the applications of majorization minimization and increment optimization. A specific surrogate function is derived using the Cauchy-Schwartz inequality, and we provide guarantees that it leads to a decrease in the original optimization problem.

The algorithm targets a high-quality facial animation for the video games and movie industry, and hence it assumes that the higher mesh fidelity is more important than the computation speed. For this reason, we rely on the quadratic blendshape function that is more precise than the standard linear one, and also that the weights are strictly constrained to $[0,1]$ interval to avoid exaggerated or non-credible expressions.

While this paper had a purpose of giving a complete derivation of the proposed algorithm, [20] presents an in-depth discussion on the applications and results.

Algorithm 2 Proposed Method

```
Require: Blendshape matrix $\mathbf{B} \in \mathbb{R}^{3n\times m}$, corrective
    blendshapes $\mathbf{b}^{\{i,j\}} \in \mathbb{R}^{3n}$ for $(i,j) \in \mathscr{P}$, target mesh
    $\widehat{\mathbf{b}} \in \mathbb{R}^{3n}$, regularization parameter $\alpha \geq 0$, initial weight
    vector $\mathbf{w}_{(0)} \in [0,1]^m$, maximum number of iterations $T \in \mathbb{N}$,
    tolerance $\varepsilon > 0$.
Ensure: $\hat{\mathbf{w}}$ - an approximate minimizer of the problem (6).
    For each coordinate $i = 1,...,3n$ compose a matrix $\mathbf{D}^{(i)} \in$
    $\mathbb{R}^{m\times m}$ from the corrective terms, and extract singular and
    eigen values $(\sigma, \lambda_{min}, \lambda_{max})$:
    for $i = 1,...,3n$ do
        for $(j,k) \in \mathscr{P}$ do
            $D^{(i)}_{jk} = D^{(i)}_{kj} = 1/2 b^{\{j,k\}}_i$.
        end for
        $\mathbf{D}^{(i)} \to \lambda_{\min}(\mathbf{D}^{(i)})$, $\lambda_{\max}(\mathbf{D}^{(i)})$, $\sigma(\mathbf{D}^{(i)})$.
    end for
    Repeat Algorithm 1 until convergence:
    for $t = 0,...,T$ do
        Compute optimal increment $\hat{\mathbf{v}}$ using Algorithm 1
        Update the weight vector $\mathbf{w}_{(t)}$:
        $\mathbf{w}_{(t+1)} = \mathbf{w}_{(t)} + \hat{\mathbf{v}}$
        Check convergence
        if $|\psi(\hat{\mathbf{v}}) - \psi(\mathbf{0})| < \varepsilon$ then
            $\hat{\mathbf{w}} \leftarrow \mathbf{w}_{(t+1)}$
            return $\hat{\mathbf{w}}$
        end if
    end for
    $\hat{\mathbf{w}} \leftarrow \mathbf{w}_{(t+1)}$
    return $\hat{\mathbf{w}}$
```

REFERENCES

[1] Bailey, S. W., Omens, D., Dilorenzo, P., O'Brien, J. F.: Fast and Deep Facial Deformations. ACM Trans. Graph. (2020) `https://doi.org/10.1145/3386569.3392397`

[2] Becker, M. P., Yang, I., Lange, K.: EM algorithms without missing data. Stat Methods Med Res. (1997) `https://doi.org/10.1177/096228029700600104`

[3] Bouaziz, S., Wang, Y., Pauly, M.: Online Modeling for Realtime Facial Animation. ACM Trans. Graph. (2013) `https://doi.org/10.1145/2461912.2461976`

[4] Çetinaslan, C. O.: Position Manipulation Techniques for Facial Animation. Porto, Portugal (2020)

[5] Cetinaslan, O., Orvalho, V.: Sketching Manipulators for Localized Blendshape Editing. Graphical Models. (2020) `https://doi.org/10.1016/j.gmod.2020.101059`

[6] Choe, B., Ko, H.S.: Analysis and synthesis of facial expressions with hand-generated muscle actuation basis. ACM SIGGRAPH 2005 Courses, (2005). `https://doi.org/10.1145/1198555.1198595`

[7] Deng, Z., Chiang, P.Y., Fox, P., Neumann, U.: Animating Blendshape Faces by Cross-Mapping Motion Capture Data. Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games. (2006) `https://doi.org/10.1145/1111411.1111419`

[8] Fang, Z., Cai, L., Wang, G.: MetaHuman Creator The starting point of the metaverse. 2021 International Symposium on Computer Technology and Information Science (ISCTIS). (2021) `https://doi.org/10.1109/ISCTIS51085.2021.00040`

[9] Feng, W.W., Kim, B.U., Yu, Y.: Real-Time Data Driven Deformation Using Kernel Canonical Correlation Analysis. ACM Trans. Graph. (2008) `https://doi.org/10.1145/1360612.1360690`

[10] Holden, D., Saito, J., Komura, T.: Learning an inverse rig mapping for character animation. Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation. (2015) `https://doi.org/10.1145/2786784.2786788`

[11] Holden, D., Saito, J., Komura, T.: Learning Inverse Rig Mappings by Nonlinear Regression. IEEE Transactions on Visualization and Computer Graphics. (2016) `https://doi.org/10.1109/TVCG.2016.2628036`

[12] Hunter, D. R., Lange, K.: A tutorial on MM algorithms. The American Statistician. (2004) https://doi.org/10.1198/0003130042836

[13] Seonghyeon, K., Sunjin, J., Kwanggyoon, S., Roger, B.R., Junyong, N.: Deep Learning-Based Unsupervised Human Facial Retargeting. Computer Graphics Forum. (2021) https://doi.org/10.1111/cgf.14400

[14] Lange, K., Hunter, D. R., Yang, I.: Optimization transfer using surrogate objective functions. Journal of Computational and Graphical Statistics. (2000) https://doi.org/10.1080/10618600.2000.10474858

[15] Lewis, J. P., Anjyo, K., Rhee, T., Zhang, M., Pighin, F. H., Deng, Z.: Practice and Theory of Blendshape Facial Models. Eurographics 2014 - State of the Art Reports. (2014) https://doi.org/10.2312/egst.20141042

[16] Li, H., Weise, T., Pauly, M.: Example-based facial rigging. ACM Trans. Graph. (2010) https://doi.org/10.1145/1778765.1778769

[17] Li, H., Yu, J., Ye, Y., Bregler, C.: Realtime facial animation with on-the-fly correctives. ACM Trans. Graph. (2013) https://doi.org/10.1145/2461912.2462019

[18] Neumann, T., Varanasi, K., Wenger, S., Wacker, M., Magnor, M., Theobalt, C.: Sparse localized deformation components. ACM Trans. Graph. (2013) https://doi.org/10.1145/2508363.2508417

[19] Porcelli, M.: On the convergence of an inexact Gauss–Newton trust-region method for nonlinear least-squares problems with simple bounds. Optimization Letters. (2013) https://doi.org/10.1007/s11590-011-0430-z

[20] Racković, S., Soares, C., Jakovetić, D., Desnica, Z: Accurate Solution of the Inverse Rig for Realistic and Complex Blendshape Models. Unpublished manuscript. Retrieved from https://gitfront.io/r/user-6651490/aa3f5ace79c1ffce557bbb12242a68f82438dab2/manuscripts/blob/ASIRRCBM.pdf

[21] Seol, Y., Lewis, J. P.: Tuning Facial Animation in a Mocap Pipeline. ACM SIGGRAPH 2014 Talks. (2014) https://doi.org/10.1145/2614106.2614108

[22] Seol, Y., Seo, J., Kim, P.H., Lewis, J. P., Noh, J.: Artist Friendly Facial Animation Retargeting. Proceedings of the 2011 SIGGRAPH Asia Conference. (2011) https://doi.org/10.1145/2024156.2024196

[23] Sifakis, E., Neverov, I., Fedkiw, R.: Automatic Determination of Facial Muscle Activations from Sparse Motion Capture Marker Data. ACM SIGGRAPH 2005 Papers. (2005) https://doi.org/10.1145/1186822.1073208

[24] Song, J., Choi, B., Seol, Y., Noh, J.Y.: Characteristic facial retargeting. ACM Trans. Graph. (2011) https://doi.org/10.1002/cav.414

[25] Song, J., Blanco, R.R., Cho, K., You, M., Lewis, J.P., Choi, B., Noh, J.: Sparse Rig Parameter Optimization for Character Animation. Computer Graphics Forum. (2017) https://dl.acm.org/doi/10.5555/3128975.3128985

[26] Song, S. L., Shi, W., Reed, M.: Accurate Face Rig Approximation with Deep Differential Subspace Reconstruction. ACM Trans. Graph. (2020) https://doi.org/10.1145/3386569.3392491

[27] Tarzanagh, D. A., Saeidian, Z., Peyghami, M. R., Mesgarani, H.: A new trust region method for solving least-square transformation of system of equalities and inequalities. Optimization Letters. (2015) https://doi.org/10.1007/s11590-013-0711-9

[28] Wang, M., Bradley, D., Zafeiriou, S., Beeler, T.: Facial Expression Synthesis using a Global-Local Multilinear Framework. Computer Graphics Forum. 39, 235-245 (2020)

[29] Yu, H., Liu, H.: Regression-based facial expression optimization. IEEE Transactions on Human-Machine Systems. (2014) https://doi.org/10.1109/THMS.2014.2313912

[30] Zhang, J., Chen, K., Zheng, J.: Facial expression retargeting from human to avatar made easy. IEEE Transactions on Visualization and Computer Graphics. (2022) https://doi.org/10.1109/TVCG.2020.3013876

[31] Zhang, Z., Kwok, J., T. Yeung, D.Y.: Surrogate maximization/minimization algorithms and extensions. Mach Learn. (2007) https://doi.org/10.1007/s10994-007-5022-x