Optimization Letters (2024) 18:545–559
https://doi.org/10.1007/s11590-023-02012-w

ORIGINAL PAPER

# A majorization–minimization-based method for nonconvex inverse rig problems in facial animation: algorithm derivation

**Stevo Racković[1] · Cláudia Soares[2] · Dušan Jakovetić[3] · Zoranka Desnica[4]**

Received: 9 May 2022 / Accepted: 4 May 2023 / Published online: 20 May 2023


## 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 that produce 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 detail for modern realistic human face reconstruction. A second-order blendshape approximation leads to higher fidelity facial animation but generates a non-linear least squares optimization problem with high dimensionality. We derive a method for solving the inverse rig in blendshape animation using quadratic corrective terms, which increases accuracy. At the same time, due to the proposed construction of the objective function, it yields a 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 and employs a Majorization–Minimization paradigm to cope with the increased complexity produced by adding 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 (Racković et al. arxiv preprint. https://arxiv.org/abs/2302.04843, 2023) 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.

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, from FCT IP strategic project NOVA LINCS (FCT UIDB/04516/2020) and project DSAIPA/AI/0087/2018. 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) and Provincial Secretariat for Higher Education and Scientific Research (Grant No. 142-451-2593/2021-01/2).

Extended author information available on the last page of the article

**Keywords** Inverse rig · Quadratic blendshape model · Majorization–Minimization

## 1 Introduction

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 [1, 2]—blendshapes are a vector representation of the face, where each vector $\mathbf{b}_1, \ldots, \mathbf{b}_m \in \mathbb{R}^{3n}$ stands for a single atomic deformation of the face, and a resting face is represented by $\mathbf{b}_0 \in \mathbb{R}^{3n}$. More complex expressions are obtained by combining the blendshape vectors, commonly using a linear delta blendshape function:

$$f(w_1, \ldots, w_m) = \mathbf{b}_0 + \sum_{i=1}^{m} w_i \Delta\mathbf{b}_i, \tag{1}$$

where $\Delta\mathbf{b}_i = \mathbf{b}_i - \mathbf{b}_0$ for $i = 1, ..., m$, and $w_i \in [0, 1]$ are activation weights corresponding to each blendshape. These models provide intuitive controls, even though building the base shapes is demanding in terms of time and effort [3–6]. Inverse rig estimation is a common problem that consists of choosing the right activation weights $w_1, \ldots, w_m$ from (1) to produce a predefined expression $\widehat{\mathbf{b}} \in \mathbb{R}^{3n}$. It is one of the aspects of blendshape animation that can be automated, hence it is often addressed in the literature, usually posed in a least-squares fashion as

$$\underset{w_1,\ldots,w_m}{\text{minimize}} \ \|f(w_1, \ldots, w_m) - \widehat{\mathbf{b}}\|^2, \tag{2}$$

with possibly additional constraints or regularization. Here, and throughout the paper, $\|\cdot\|$ stands for the $l_2$ norm. 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 excellent performance [7–17], 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 [2, 18–20] 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 a 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 [7, 21]. Our method utilizes a common framework of Majorization–Minimization [22, 23].

### 1.1 Contributions

In the companion paper [24], 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 [24] 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 2 formulates the inverse rig problem and covers the existing solutions from the literature. Section 3 introduces our algorithm and gives a detailed derivation of each step. Section 4 discusses numerical results. Finally, we conclude the paper in Sect. 5.

## 2 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 $\mathbf{b}_1, \ldots, \mathbf{b}_m \in \mathbb{R}^{3n}$, 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 $i = 1, \ldots, m$, 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 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 can then be defined as

$$f_L(\mathbf{w}) = \mathbf{b}_0 + \sum_{i=1}^{m} w_i\Delta\mathbf{b}_i = \mathbf{b}_0 + \mathbf{B}\mathbf{w}, \tag{3}$$

where $\mathbf{w} = [w_1, \ldots, w_m]^T$ is a weight vector and $\mathbf{B} = [\Delta\mathbf{b}_1, \ldots, \Delta\mathbf{b}_m]$ is a blendshape matrix. The notation $f_L(\cdot)$ 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.

Some pairs of blendshapes, $\mathbf{b}_i$ and $\mathbf{b}_j$, with an 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 and make the character appearance natural. An artist discovers these combinations, and corrective terms are conventionally sculpted by hand. 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. We define a set $\mathcal{P}$ that stores tuples of indices $(i, j)$ such that a pair of belndshapes $i$ and $j$ have a corresponding corrective term $\mathbf{b}^{\{i,j\}}$. A quadratic blendshape function is then defined as:

$$f_Q(\mathbf{w}) = b_0 + \sum_{i=1}^{m} w_i \Delta\mathbf{b}_i + \sum_{(i,j)\in\mathcal{P}} w_i w_j \mathbf{b}^{\{i,j\}}. \tag{4}$$

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.

### 2.1 Existing solutions

When we consider a model-based solution to the inverse rig problem, the state-of-the-art method is [25], where the optimization problem is formulated as regularized least-squares minimization:

$$\underset{\mathbf{w}}{\text{minimize}} \ \|\mathbf{B}\mathbf{w} - \widehat{\mathbf{b}}\|^2 + \alpha\|\mathbf{w}\|^2. \tag{5}$$

Here, and throughout the paper, $\|\cdot\|$ stands for the $l_2$ norm. Regularization is necessary because many blendshape pairs are highly correlated, i.e., produce relatively similar deformations over the mesh; hence, the unregularized problem is often ill-posed, and the solution is not unique. Additionally, it is desired to keep the number of non-zero elements of $\mathbf{w}$ low because it allows for further manual editing, which is common in animation. The solution to (5) is given in a closed-form as

$$\mathbf{w} = (\mathbf{B}^T\mathbf{B} + \alpha\mathbf{I})^{-1}\mathbf{B}^T\widehat{\mathbf{b}}. \tag{6}$$

A modification to this solution is given in the same paper. Authors approximate a blendshape matrix $\mathbf{B}$ with a sparse matrix $\mathbf{B}^{loc}$, 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 [26], where components are visited and optimized sequentially, and after each iteration $i = 1, \ldots, m$, a residual term is updated:

$$\begin{aligned} &\text{step 1:} \quad \underset{w_i}{\text{minimize}} \ \|w_i\Delta\mathbf{b}_i - \widehat{\mathbf{b}}\|^2, \\ &\text{step 2:} \quad \widehat{\mathbf{b}} \leftarrow \widehat{\mathbf{b}} - w_i\Delta\mathbf{b}_i. \end{aligned} \tag{7}$$

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 (5) or (7), 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 a realistic human face representation in high-quality movies and video games. In the next section, we introduce our solution to the inverse rig problem that takes into account quadratic corrective terms.

## 3 Proposed solution

This section presents a detailed derivation of our method that utilizes second-order blendshape models; we refer to [24] for a 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 bounded to [0, 1] interval[1]. We first explain the algorithm derivation and then detail each algorithm step. The optimization problem looks for the optimal weight vector configuration $\mathbf{w}$ that fits on a given mesh $\widehat{\mathbf{b}}$, assuming a quadratic blendshape function (4), as

$$\underset{\mathbf{0}\leq\mathbf{w}\leq\mathbf{1}}{\text{minimize}} \ \left\|f_Q(\mathbf{w}) - \widehat{\mathbf{b}}\right\|^2 + \alpha\mathbf{1}^T\mathbf{w}, \tag{8}$$

where the first term is data fidelity and the second is regularization. The non-negativity constraint is important in blendshape animation since negative weights have no semantical meaning and make it harder for animators to adjust the obtained results manually. While weights larger than one might be useful for exaggerated cartoonish expressions, in realistic human avatars this is not a favorable behavior. The regularization term with $\alpha > 0$ enforces a low cardinality of the solution vector, which is a desired feature as it makes the results more artist-friendly [26]. The problem is approached in a fashion similar to the Levenberg-Marquardt method [27], where we choose the initial vector of controller weights $\mathbf{w}_{(0)} \in \mathbb{R}^m$, and at each iteration $t = 0, \ldots, T$, solve for an optimal increment vector $\mathbf{v} \in \mathbb{R}^m$ that solves the following optimization problem:

$$\underset{-\mathbf{w}_{(t)}\leq\mathbf{v}\leq\mathbf{1}-\mathbf{w}_{(t)}}{\text{minimize}} \ \left\|f_Q(\mathbf{w}_{(t)} + \mathbf{v}) - \widehat{\mathbf{b}}\right\|^2 + \alpha\mathbf{1}^T(\mathbf{w}_{(t)} + \mathbf{v}). \tag{9}$$

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

[1] 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.

objective function in (9) is fairly complex, hence we simplify it by applying Majorization–Minimization (MM). That is, to solve (9), we define an upper bound function $\psi(\mathbf{v};\mathbf{w}) : \mathbb{R}^m \to \mathbb{R}$ over the objective function in (9) 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 (9), 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:

$$\|f_Q(\mathbf{w}+\mathbf{v}) - \widehat{\mathbf{b}}\|^2 + \alpha \mathbf{1}^T(\mathbf{w} + \mathbf{v}) \leq \psi(\mathbf{v};\mathbf{w}). \tag{10}$$

**Condition 2** The upper bound $\psi(\mathbf{v};\mathbf{w})$ matches the value of the objective (9) at point $\mathbf{v} = \mathbf{0}$, i.e.,

$$\|f_Q(\mathbf{w}) - \widehat{\mathbf{b}}\|^2 + \alpha \mathbf{1}^T(\mathbf{w}) = \psi(\mathbf{0};\mathbf{w}). \tag{11}$$

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

$$\underset{-\mathbf{w}\leq\mathbf{v}\leq\mathbf{1}-\mathbf{w}}{\text{minimize}}\ \psi(\mathbf{v}), \tag{12}$$

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* (12) (*with* $\mathbf{w}_{(t+1)} = \mathbf{w}_{(t)} + \mathbf{v}_{(t)}$) *produces a monotonically non-increasing sequence of values of the objective* (8), *i.e.*,

$$\|f_Q(\mathbf{w}_{(t)}) - \widehat{\mathbf{b}}\|^2 + \alpha \mathbf{1}^T\mathbf{w}_{(t)} \geq \|f_Q(\mathbf{w}_{(t+1)}) - \widehat{\mathbf{b}}\|^2 + \alpha \mathbf{1}^T\mathbf{w}_{(t+1)}\ ,\ t \in \mathbb{N}. \tag{13}$$

***Proof*** If $\mathbf{v}_{(t)}$ is a minimizer of (12) at iteration $t$, i.e.,

$$\mathbf{v}_{(t)} = \underset{-\mathbf{w}_{(t)}\leq\mathbf{v}\leq\mathbf{1}-\mathbf{w}_{(t)}}{\text{argmin}}\ \psi(\mathbf{v}),$$

then we have $\psi(\mathbf{0}) \geq \psi(\mathbf{v}_{(t)})$. From this and from *Conditions* 2 and 1, we have the following relation:

$$\begin{aligned} &\|f_Q(\mathbf{w}_{(t)}) - \widehat{\mathbf{b}}\|^2 + \alpha \mathbf{1}^T(\mathbf{w}_{(t)}) \\ &\quad = \psi(\mathbf{0}) \geq \psi(\mathbf{v}_{(t)}) \\ &\quad \geq \|f_Q(\mathbf{w}_{(t)} + \mathbf{v}_{(t)}) - \widehat{\mathbf{b}}\|^2 + \alpha \mathbf{1}^T(\mathbf{w}_{(t)} + \mathbf{v}_{(t)}), \end{aligned} \tag{14}$$

which proves the proposition. □

Additionally, from [28], we have that under the above iterative method, the objective function converges to a local optimum or a saddle point as number of iterations $t$ goes to infinity.

*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 objective (9) as

$$\psi(\mathbf{v}) = \sum_{i=1}^{3n} \psi_i(\mathbf{v}) + \alpha \mathbf{1}^T(\mathbf{w} + \mathbf{v}), \tag{15}$$

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 the number of vertices in the mesh. The component-wise bounds have the form

$$\psi_i(\mathbf{v}) := g_i^2 + 2g_i \sum_{j=1}^{m} h_{ij} v_j + 2\big(g_i \lambda_M(\mathbf{D}^{(i)}, g_i) + \|\mathbf{h}_i\|^2\big) \sum_{j=1}^{m} v_j^2 + 2m\sigma^2(\mathbf{D}^{(i)}) \sum_{j=1}^{m} v_j^4 \tag{16}$$

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 (15), the problem (12) can be analytically solved component-wise, where for each component $j = 1, \ldots, m$, the objective is a scalar quartic equation:

$$\begin{aligned} &\underset{v_j}{\text{minimize}} \quad q v_j + r v_j^2 + s v_j^4, \\ &\text{s.t.} \quad 0 \leq w_j + v_j \leq 1. \end{aligned} \tag{17}$$

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

$$q = 2 \sum_{i=1}^{3n} g_i h_{ij} + \alpha, \quad r = 2 \sum_{i=1}^{3n} (g_i \lambda_M(\mathbf{D}^{(i)}, g_i) + \|\mathbf{h}_i\|^2), \quad s = 2m \sum_{i=1}^{3n} \sigma^2(\mathbf{D}^{(i)}). \tag{18}$$

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, if they are within the feasible interval [0, 1], compare them 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 [26], 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 [26] 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 (9), in a manner similar to the trust-region method [29, 30]. 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 (15).

*Derivation of the Upper Bound.* If we write down the data fidelity term from (9) as a sum, and consider each element $i = 1, \ldots, 3n$ of the sum separately, we can represent it in a canonical quadratic form:

$$\sum_{i=1}^{3n} \left( \mathbf{B}_i \mathbf{w} + \sum_{(j,k)\in\mathcal{P}} w_j w_k b_i^{\{j,k\}} - \widehat{b}_i \right)^2 = \sum_{i=1}^{3n} \left( \mathbf{B}_i \mathbf{w} + \mathbf{w}^T \mathbf{D}^{(i)} \mathbf{w} - \widehat{b}_i \right)^2. \tag{19}$$

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:

$$\begin{aligned} \phi_i(\mathbf{w} + \mathbf{v}) &= \left( g_i + \mathbf{h}_i \mathbf{v} + \mathbf{v}^T \mathbf{D}^{(i)} \mathbf{v} \right)^2 \\ &= g_i^2 + 2 g_i \mathbf{h}_i \mathbf{v} + 2 g_i \mathbf{v}^T \mathbf{D}^{(i)} \mathbf{v} + \left( \mathbf{h}_i \mathbf{v} + \mathbf{v}^T \mathbf{D}^{(i)} \mathbf{v} \right)^2. \end{aligned} \tag{20}$$

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 $\phi_i(\mathbf{w} + \mathbf{v}) \leq \psi_i(\mathbf{v})$. Let us first consider two nonlinear terms of $\phi_i(\mathbf{w})$, and bound each of them separately. A bound over the quadratic term $2 g_i \mathbf{v}^T \mathbf{D}^{(i)} \mathbf{v}$ depends on the sign of $g_i$:

$$2 g_i \mathbf{v}^T \mathbf{D}^{(i)} \mathbf{v} \leq 2 g_i \lambda_M \left( \mathbf{D}^{(i)}, g_i \right) \|\mathbf{v}\|^2. \tag{21}$$

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

$$\begin{aligned} &(\mathbf{h}_i \mathbf{v} + \mathbf{v}^T \mathbf{D}^{(i)} \mathbf{v})^2 \leq 2(\mathbf{h}_i \mathbf{v})^2 + 2(\mathbf{v}^T \mathbf{D}^{(i)} \mathbf{v})^2 \\ &\leq 2\|\mathbf{h}_i\|^2 \|\mathbf{v}\|^2 + 2\|\mathbf{v}\|^4 \|\mathbf{D}^{(i)}\|^2 \leq 2\|\mathbf{h}_i\|^2 \|\mathbf{v}\|^2 + 2m\sigma^2(\mathbf{D}^{(i)}) \sum_{j=1}^{m} v_j^4. \end{aligned} \tag{22}$$

The component-wise bound $\psi_i(\mathbf{v})$ is then given by (16) and a complete upper bound is obtained by summing the component-wise bounds and adding the regularization term (Eq. 15).

**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 $\alpha \geq 0$ and weight vector $\mathbf{w} \in [0, 1]^m$.

**Ensure:** $\hat{\mathbf{v}}$ - an optimal increment vector as a solution to (17).

Compute coefficients $q, r, s$ from Eq. (18) 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**
  $q = 2\sum_{i=1}^{3n} g_i h_{ij} + \alpha$
  $\hat{v}_k = \operatorname{argmin}_v\, qv + rv^2 + sv^4$
    s.t. $-w_k \leq v \leq 1 - w_k$
**end for**
**return** $\hat{\mathbf{v}}$

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

**Proposition 2** *The surrogate function* $\psi(\mathbf{v}) : \mathbb{R}^m \to \mathbb{R}$, *defined as in* (15) *satisfies Conditions* 1 *and* 2, *and hence it satisfies Proposition* 1.

In (20)-(22), function $\psi(\cdot)$ is derived so that it bounds the objective function (19) 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 (20) we see that $\phi_i(\mathbf{w} + \mathbf{0}) = g_i^2$, and from (16) we get $\psi_i(\mathbf{0}) = g_i^2$, which proves it. Hence, the derived bound satisfies *Proposition* 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 $\epsilon > 0$. While any feasible vector $\mathbf{0} \leq \mathbf{w} \leq \mathbf{1}$ 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 [24]. 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.

**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 $(i, j) \in \mathcal{P}$, 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 $\epsilon > 0$.
**Ensure:** $\hat{\mathbf{w}}$ - an approximate minimizer of the problem (8).
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** $(j, k) \in \mathcal{P}$ **do**
    $D_{jk}^{(i)} = D_{kj}^{(i)} = 1/2b_i^{\{j,k\}}$.
  **end for**
  $\mathbf{D}^{(i)} \rightarrow \lambda_{\min}(\mathbf{D}^{(i)})$, $\lambda_{\max}(\mathbf{D}^{(i)})$, $\sigma(\mathbf{D}^{(i)})$.
**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** $|\psi(\hat{\mathbf{v}}) - \psi(\mathbf{0})| < \epsilon$ **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** $\hat{\mathbf{w}}$

**Corollary 1** *The estimate sequence* $\mathbf{w}_{(t)}$ *produced by Algorithm* 2 *is feasible for problem* (8) *at all iterations* $t \in \mathbb{N}$, *as long as the initial weight vector* $\mathbf{w}_{(0)}$ *is feasible*.

## 4 Results

The experiments in this section are performed over an animated avatar *Omar*, available at Metahuman Creator.[2] The character consists of $m = 130$ base blendshapes and $n = 4000$ vertices in the face. An optimized choice of regularization parameter $\alpha = 5$ is estimated experimentally from the training data, to give a good trade-off between the high accuracy of the mesh fit and a low cardinality of the weight vector. In our experiments, the weight vector is initialized by (6), where the weights outside of the feasible set are clipped to 0 or 1, and then further optimized using our algorithm. While any feasible vector can be used for initialization, this choice has shown quick convergence and precise mesh fit.

[2] unrealengine.com/en-US/metahuman.

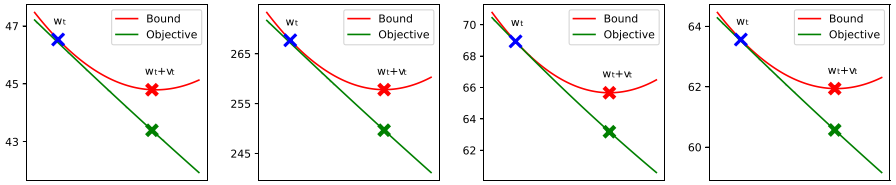


**Fig. 1** Several example frames showing a relationship between the objective function (green) and the proposed surrogate function (red), at iteration $t$. A current iterate $\mathbf{w}_t$ is marked by a blue cross, and the estimated next iterate $\mathbf{w}_t + \mathbf{v}_t$ is marked by a red cross showing the values of a surrogate function, and by a green one, indicating the actual value of the objective function

Let us first examine the relationship between the objective function (8) and its corresponding surrogate function (15). Figure 1 shows several example frames and the behavior of two functions in the initial iterations. The blue cross represents the value of the function at an initial iterate $\mathbf{w}_t$, i.e., for $\mathbf{v} = 0$ (As stated in *Condition* 2, the two functions have the same value in this point). Values of the two functions are then denoted by the corresponding color cross in the estimated iterate $\mathbf{w}_t + \mathbf{v}_t$. The other values of the two curves are obtained by interpolating the two vectors $\mathbf{w}_t$ and $\mathbf{w}_t + \mathbf{v}_t$ and estimating the corresponding function value. These results clearly show how minimizing the upper bound leads to a nice decrease in the objective.

Further, we show the final results of the method in Fig. 2. On the left side of the subfigures, one can see a barplot representing the estimated activation weights in the range [0,1]—we can see that the vector is relatively sparse, and most of the weights have low values. The average cardinality of the weight vector, over 500 test frames, is 85. Below we have the behavior of the objective function over iterations of the algorithm. Green crosses represent the value of the function (8) for the estimated iterate $\mathbf{w}_t + \mathbf{v}_t$, while red pluses denote the value of the corresponding upper bound function (15), in line with the results presented in Fig. 1. The plot zooms on initial iterations to show how the proposed bound is tight, and the gap between the surrogate and objective quickly decreases. We can also notice a nice decrease in the objective, confirming the above-stated monotonicity and convergence properties.

Finally, on the right side of the subfigures, we show the reference mesh $\widehat{\mathbf{b}}$ in gray, versus the reconstruction obtained by the proposed method. The reconstructed mesh is colored so that red tones indicate a higher error, according to the color bar on the right. We can see that in each case, the reconstruction is really close to the original frame, and even the regions indicated by the red color are not visually different. The average root mean squared error over 500 test frames is 0.09.

For a detailed numerical discussion and comparison with other methods see [24]—the reference examines results over multiple animated characters and shows that the proposed method produces accurate and sparse solutions to the inverse rig problem.

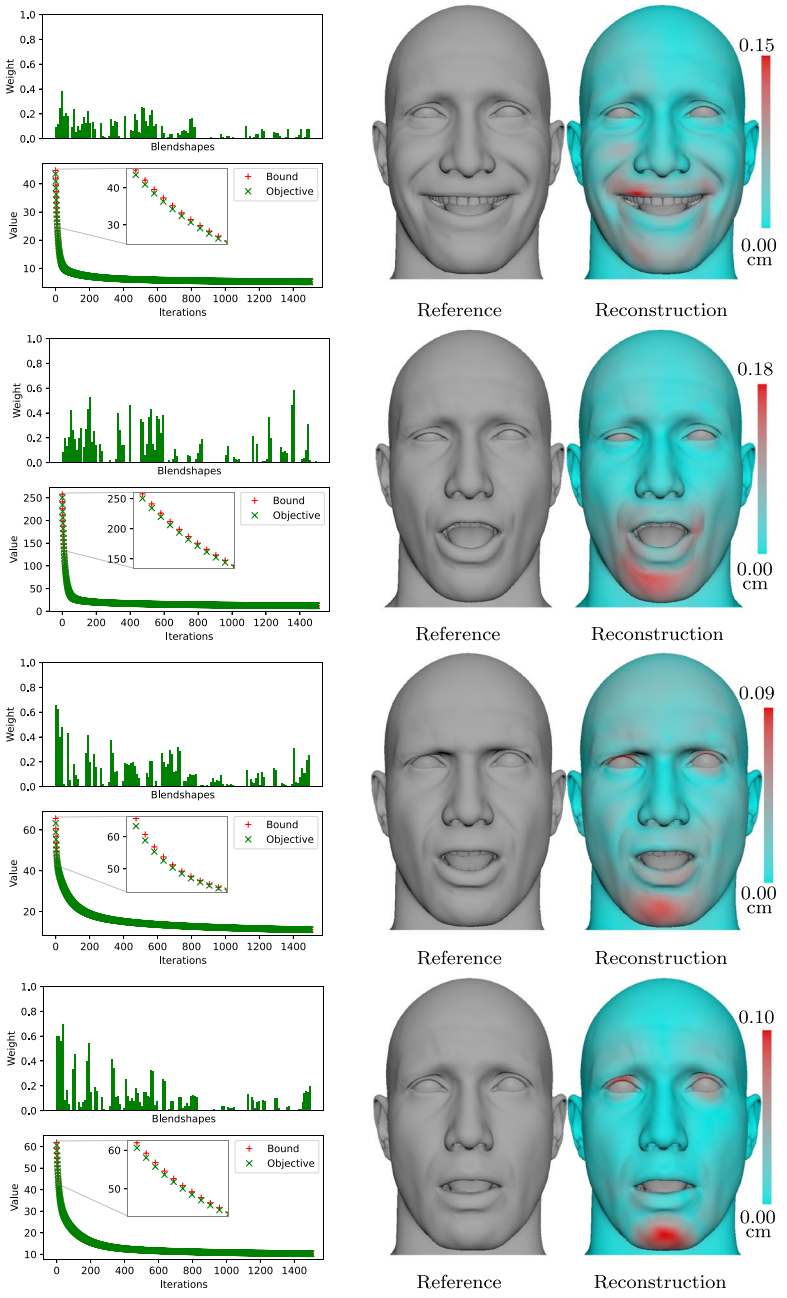


**Fig. 2** Results over four animated frames. Barplots on the left show the estimated activation weight vector **w**; below that is a scatter-plot showing the behavior of the objective function and the upper bound over the iterations; on the right, we have a reference mesh (gray) and the reconstruction by the proposed algorithm—colors indicate the level of offset over the mesh, corresponding to the color-bar on the right

## 5 Conclusion

This paper introduced the first model-based method for solving the inverse rig problem under the quadratic blendshape function. The proposed algorithm is based on the applications of Majorization–Minimization. A specific surrogate function is derived, and we provide guarantees that it leads to a non-increasing cost sequence of the original non-convex optimization problem (8).

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 a [0, 1] interval to avoid exaggerated or non-credible expressions.

While this paper had the purpose of giving a complete derivation of the proposed algorithm, [24] presents an in-depth discussion of the applications and results.

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1007/s11590-023-02012-w.

**Funding** Open access funding provided by FCT|FCCN (b-on).



## References

1. 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
2. Çetinaslan, C.O.: Position Manipulation Techniques for Facial Animation. Porto, Portugal (2020)
3. 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
4. Wang, M., Bradley, D., Zafeiriou, S., Beeler, T.: Facial expression synthesis using a global-local multilinear framework. Comput. Graph. Forum. **39**, 235–245 (2020)
5. Li, H., Yu, J., Ye, Y., Bregler, C.: Realtime facial animation with on-the-fly correctives. ACM TOG (2013). https://doi.org/10.1145/2461912.2462019
6. Zhang, J., Chen, K., Zheng, J.: Facial expression retargeting from human to avatar made easy. IEEE Trans. Visual Comput. Graph. (2022). https://doi.org/10.1109/TVCG.2020.3013876
7. Holden, D., Saito, J., Komura, T.: Learning inverse rig mappings by nonlinear regression. IEEE Trans. Vis. Comput. Graph. (2016). https://doi.org/10.1109/TVCG.2016.2628036
8. Bailey, S.W., Omens, D., Diloreno, P., O'Brien, J.F.: Fast and deep facial deformations. ACM TOG (2020). https://doi.org/10.1145/3386569.3392397

9. Song, S.L., Shi, W., Reed, M.: Accurate face rig approximation with deep differential subspace reconstruction. ACM TOG (2020). https://doi.org/10.1145/3386569.3392491
10. Seonghyeon, K., Sunjin, J., Kwanggyoon, S., Roger, B.R., Junyong, N.: Deep learning-based unsupervised human facial retargeting. Comput. Graph. Forum. (2021). https://doi.org/10.1111/cgf.14400
11. 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
12. Song, J., Choi, B., Seol, Y., Noh, J.Y.: Characteristic facial retargeting. ACM TOG (2011). https://doi.org/10.1002/cav.414
13. Seol, Y., Lewis, J.P.: Tuning facial animation in a mocap pipeline. ACM SIGGRAPH 2014 Talks. (2014) https://doi.org/10.1145/2614106.2614108
14. 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
15. Feng, W.W., Kim, B.U., Yu, Y.: Real-time data driven deformation using kernel canonical correlation analysis. ACM TOG (2008). https://doi.org/10.1145/1360612.1360690
16. Yu, H., Liu, H.: Regression-based facial expression optimization. IEEE Trans. Human-Mach. Syst. (2014). https://doi.org/10.1109/THMS.2014.2313912
17. Bouaziz, S., Wang, Y., Pauly, M.: Online modeling for realtime facial animation. ACM TOG (2013). https://doi.org/10.1145/2461912.2461976
18. 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
19. 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
20. Li, H., Weise, T., Pauly, M.: Example-based facial rigging. ACM TOG (2010). https://doi.org/10.1145/1778765.1778769
21. Song, J., Blanco, R.R., Cho, K., You, M., Lewis, J.P., Choi, B., Noh, J.: Sparse rig parameter optimization for character animation. Comput. Graph. Forum. (2017). https://doi.org/10.5555/3128975.3128985
22. 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
23. Lange, K., Hunter, D.R., Yang, I.: Optimization transfer using surrogate objective functions. J. Comput. Graph. Stat. (2000). https://doi.org/10.1080/10618600.2000.10474858
24. Racković, S., Soares, C., Jakovetić, D., Desnica, Z: Accurate and Interpretable Solution of the Inverse Rig for Realistic Blendshape Models with Quadratic Corrective Terms. Unpublished manuscript. Retrieved from https://github.com/stevorackovic/manuscripts/blob/main/Rackovic22AcurateInterpretable.pdf
25. Cetinaslan, O., Orvalho, V.: Sketching manipulators for localized blendshape editing. Graph. Models (2020). https://doi.org/10.1016/j.gmod.2020.101059
26. 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
27. Ranganathan, A.: The Levenberg-Marquardt algorithm. Tutoral LM Algorithm. **11**(1), 101–110 (2004)
28. Wu, C.F.J.: On the convergence properties of the EM algorithm. The Annals of statistics. 95-103 (1938)
29. 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. Optim. Lett. (2015). https://doi.org/10.1007/s11590-013-0711-9
30. Porcelli, M.: On the convergence of an inexact Gauss-Newton trust-region method for nonlinear least-squares problems with simple bounds. Optim. Lett. (2013). https://doi.org/10.1007/s11590-011-0430-z

## Authors and Affiliations

**Stevo Racković[1] · Cláudia Soares[2] · Dušan Jakovetić[3] · Zoranka Desnica[4]**

✉ Stevo Racković
stevo.rackovic@tecnico.ulisboa.pt

[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, Novi Sad, Serbia