

---

# ACCURATE AND INTERPRETABLE SOLUTION OF THE INVERSE RIG FOR REALISTIC BLENDSHAPE MODELS WITH QUADRATIC CORRECTIVE TERMS

---

A PREPRINT

 Stevo Racković

Department of Mathematics  
Instituto Superior Técnico  
Lisbon, Portugal

stevo.rackovic@tecnico.ulisboa.pt

 Cláudia Sores

Department of Computer Sceince  
NOVA School of Science and Technology  
Caparica, Portugal

 Dušan Jakovetić

Department of Mathematics  
University of Novi Sad  
Novi Sad, Serbia

 Zoranka Desnica

3Lateral Animation Studio  
Epic Games Company

November 2022

## ABSTRACT

We propose a new model-based algorithm for solving the inverse rig problem in facial animation retargeting, exhibiting higher accuracy of the fit and sparser, more interpretable weight vector compared to state-of-the-art methods. The proposed method targets a specific subdomain of human face animation — highly-realistic blendshape models used in the production of movies and video games. In this paper, we formulate an optimization problem that takes into account all the requirements of targeted models. Unlike the prior solutions, our objective goes beyond a linear blendshape model and employs the quadratic corrective terms necessary for correctly fitting fine details of the mesh. Further, we show that the solution to the proposed problem yields highly accurate mesh reconstruction even when general-purpose solvers, like sequential quadratic programming, are used. The results obtained using general-purpose solvers are highly accurate in the mesh space but do not exhibit favorable qualities in terms of weight sparsity and smoothness, and for this reason, we further propose a novel algorithm relying on a majorization-minimization technique. The algorithm is specifically suited for solving the proposed objective, yielding a high-accuracy mesh fit while respecting the constraints and producing a sparse and smooth set of weights that are easy to manipulate and interpret by animation artists. We show results using both proprietary and open-source animated characters of high quality and level of detail. Our algorithm is benchmarked with state-of-the-art approaches, and shows an overall superiority of the results, yielding a smooth animation reconstruction with a relative improvement up to 45% in root mean squared mesh error while keeping the cardinality comparable with benchmark methods. This paper gives a comprehensive set of evaluation metrics that cover different aspects of the solution, including mesh accuracy, sparsity of the weights, and smoothness of the animation curves, as well as the appearance of the produced animation, which human experts evaluated.

**Keywords** Inverse Rig · Quadratic blendshape model · Majorization Minimization

## 1 Introduction

Facial animation is a growing research topic in academia as well as in industry due to the central role of facial expressions in verbal and non-verbal communication, impacting mainly the arts (animation, video games) and other

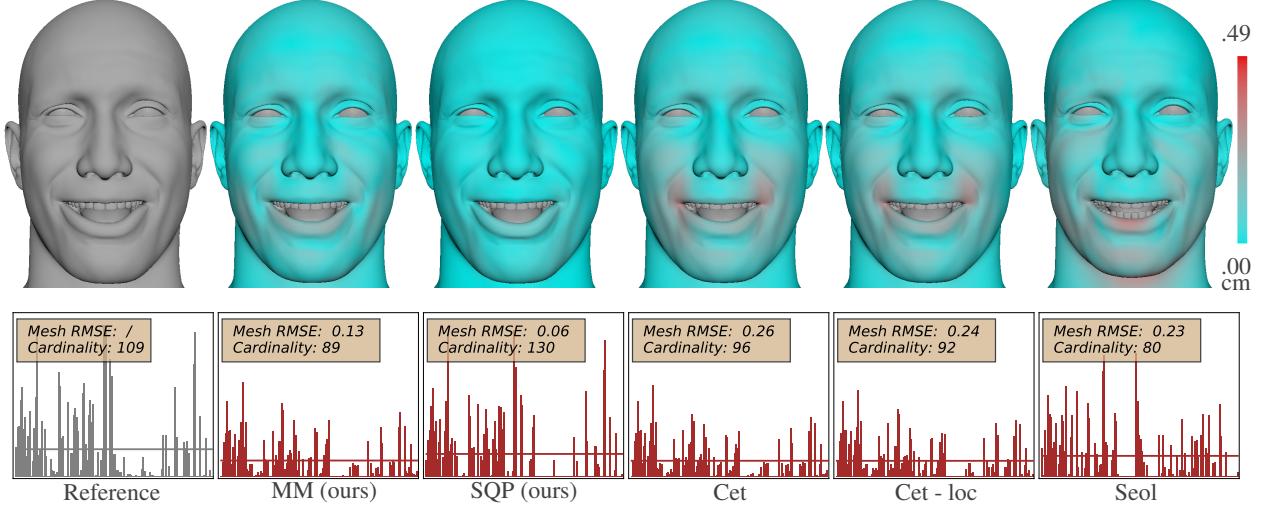


Figure 1: The reference frame mesh and the estimates of the approach that solves the proposed objective function using our novel algorithm (*MM*), the approach that solves the proposed objective applying a general-purpose solver (*SQP*), and linear approaches proposed by Cetinaslan and Orvalho [2020a] (*Cet* for the standard case, and *Cet-loc* for localized approximation) and Seol et al. [2011] (*Seol*). The top row shows obtained meshes, while the bottom represents corresponding activations of the blendshape weights. Red tones in the meshes indicate a higher error of the fit, according to the color bar on the right. The average weight activation of each solution is indicated with a horizontal line. The average mesh error and cardinality (i.e., the number of non-zero weights) of the weight vector are given in a text box for each method — we aim for the lowest error while keeping the cardinality relatively low. *SQP* yields the cleanest mesh fit of all, yet the cardinality of the weight vector is too high, hence it might compromise the stability of the solution. On the other side, linear methods *Cet*, *Cet-loc* and *Seol* give visible flaws in mesh reconstruction. *MM* is the only method that provides both an accurate mesh fit and a stable solution.

areas, like marketing (animated advertisement, video chatbots). Although there is a number of models developed to credibly deform a 3D character’s face, including the underlying anatomical structure Choe et al. [2001], Sifakis et al. [2005], Wu et al. [2016], Zoss et al. [2018, 2019], Marco et al. [2020], physics-based deformations Hahn et al. [2013], Ichim et al. [2016, 2017], or morphable models Hu et al. [2005], Smet and Gool [2010], Thies et al. [2015], Mengjiao et al. [2020], the most popular approach is the blendshape model Pighin et al. [1998], Choe and Ko [2006], Joshi et al. [2006], Deng et al. [2006], Lewis and Anjyo [2010], Li et al. [2010], Lewis et al. [2014]. Even though this approach might suffer from reduced expressivity, it provides intuitive controls and is easy to use and understand Ichim et al. [2017]. Traditionally, the (delta) blendshape model is presented as a linear mapping

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

where  $\mathbf{b}_0 \in \mathbb{R}^{3n}$  is a column vector representing the face mesh (with  $n$  vertices) in a resting position and  $\mathbf{b}_1, \dots, \mathbf{b}_m \in \mathbb{R}^{3n}$  are delta blendshape vectors, i.e., these are topologically identical copies of  $\mathbf{b}_0$ , but each of them is locally deformed, corresponding to an atomic facial expression. Blendshape vectors span the space of feasible expressions of the 3D character and can be invoked isolated or combined with other blendshapes to build more complex facial expressions. This is achieved by assigning corresponding activation weights  $w_1, \dots, w_m$  to each blendshape. These weights are scalars, and are often (but not exclusively) restricted to the  $[0, 1]$  interval. To ease the notation, blendshape weights can be concatenated into a column weight vector  $\mathbf{w} = [w_1, \dots, w_m]^T$ , and blendshape vectors can be collected to build columns of a blendshape matrix  $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_m]$ , which gives a matrix expression in (1). (More details will be given in Sec. 3.1)

Once the blendshape model is built, a character can be animated directly by setting the values of the weights for each animation frame. Since high-quality characters usually have over one hundred blendshapes, manually setting the weights for each frame involves intensive labor, and large production times. The animation can be automated if the reference motion is available in the form of 3D scans or a set of markers obtained via motion capture (MoCap). This process is called animation retargeting, and algorithms are developed to estimate the optimal set of activation weights for each frame so that the character mesh is deformed to closely resemble a reference mesh  $\hat{\mathbf{b}} \in \mathbb{R}^{3n}$  at a given frame.



Figure 2: Examples of dense activation vectors. All the activation weights of this character model are set to a random value within a feasible interval  $[0, 1]$ , and we can see that there are many anatomically incorrect deformations as well as the breaking of the mesh.

We call the problem of finding an optimal  $\mathbf{w}$  the **inverse rig problem**. This solution vector needs to satisfy several properties in order to be considered good. In the first place, we measure the data fidelity, i.e., the resemblance between a reconstructed and a reference mesh. Another essential feature is the stability of the solution, so that if the few activated weights are slightly adjusted afterwards, the mesh behaves predictably and does not produce artifacts like discontinuities or anatomically impossible positions (Figure 2). This is important because the results of the automatic retargeting are often adjusted manually in production Cetinaslan and Orvalho [2020b]. For the same reason, it is preferable to have lower cardinality, i.e., the number of non-zero values of the weight vector. If the animation retargeting is performed over a continuous time sequence, there should not be visible discontinuities between the frames. Finally, if the range of weight values is restricted by the animation framework in use (e.g.,  $0 \leq w_i \leq 1$ ), the solution needs to respect the constraints.

Our main interest in this paper is a solution to the inverse rig problem for realistic human characters (Figure 3). We make a distinction between *data-based* and *model-based* approaches for solving the inverse rig — the former assumes a rig function as a black box and demands long animation sequences that span a whole space of expected expressions in order to train a good regressor, while the latter only requires a well-defined rig function with corresponding basis vectors. The literature offers several model-based methods for solving the inverse rig under the linear blendshape model Choe and Ko [2006], Joshi et al. [2006], Lewis and Anjyo [2010], Seol et al. [2011], Liu et al. [2010], Cetinaslan and Orvalho [2020a]; however, due to the increasing complexity and level of realism of the avatars in the movie and gaming industry (but also for purposes of communication, education, virtual reality), linear models do not provide a high-enough level of detail. A possible approach is the application of data-based machine learning algorithms, yet this is an expensive alternative as it demands a large amount of data to provide a good fit.

In this paper, we propose a model-based algorithm that solves the inverse rig problem for realistic human face blendshape models used in the industry, taking into account the quadratic corrective terms of a blendshape model and the constraints over the weights vector. Our algorithm is benchmarked with state-of-the-art methods Seol et al. [2011], Cetinaslan and Orvalho [2020a], and it exhibits a significant advantage in data fidelity of the reconstructed meshes (a relative improvement up to 45% in root mean squared mesh error, computed as explained in Section 5.2) while at the same time cardinality (i.e., the number of non-zero elements) of the weight vector is comparable to previous solutions, and the frame-to-frame transitions are smooth.

## 1.1 Contributions

The contributions of this paper are two-fold.

- We address the inverse rig problem with quadratic corrective terms Seo et al. [2011], Lewis et al. [2014], and show evidence of decreasing mesh error when compared with the commonly used linear blendshape model (1). To the best of our knowledge, this paper presents the first model-based solution to the inverse rig problem under the quadratic blendshape function. We show experimentally that the new objective can be optimized using standard solvers with significantly reduced error in the mesh space compared to the state-of-the-art methods. Specifically, we use an interior-point-based solver within Python library `scipy` Virtanen et al. [2020], that is based on the sequential Quadratic Programming (SQP) applied iteratively with Trust-Region method Byrd et al. [1999], Conn et al. [2000]. However, using such a general-purpose solver empirically shows an increased number of activated blendshapes and frame-to-frame discontinuities. This leads us to our second contribution.
- This paper formulates an algorithm that is specifically suited for solving the inverse rig problem for a quadratic blendshape model. (Details of the quadratic blendshape model are given in Section 3.2.) In particular, we apply the majorization-minimization technique Sun et al. [2016], Marnissi et al. [2020] and devise a surrogate

function that allows for an efficient iterative solution to the proposed optimization problem. The obtained solution has a low mesh error, while simultaneously decreasing the number of activated blendshape weights (cardinality). Frame-to-frame transitions are shown to be smooth, and a user study with animation artists has evidenced the higher quality of our approach.

This paper presents the solution from a domain point of view, including an explanation of how the method works, a comprehensive set of experiments on proprietary and open-source state-of-the-art animation characters, a detailed modeling rationale and motivation for introducing and modeling quadratic blendshape models, the intuition behind the method and implementation details. In a companion paper not for review [2022], we present the detailed method derivation from the optimization theory perspective and convergence analysis of the algorithm.

The rest of this paper is organized as follows. In Section 2, we cover the prior work and relate our method to the corresponding directions of research. In Section 3, we cover the main concepts of the blendshape model for facial animation and the problem of rig inversion. Section 4 introduces the proposed method for solving the inverse rig, and Section 5 gives an extensive numerical evaluation of the proposed method. Finally, Section 6 concludes the paper.

## 1.2 Notation

Throughout this paper, scalar values will be denoted with lowercase Latin  $a, b, c$ , or lowercase Greek  $\alpha, \beta, \gamma$  letters. Vectors are represented with bold lowercase  $\mathbf{a}, \mathbf{b}, \mathbf{c}$  and are indexed using a subscript, i.e., the  $i^{th}$  element of the vector  $\mathbf{a}$  is  $a_i$ . If there is a subscript and the letter is still in bold, it is not indexing — we will use this to differentiate blendshape vectors ( $\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_m$ ) as they have similar properties, or to indicate that a vector  $\mathbf{a}$  takes specific value at iteration  $i$  of an iterative algorithm, which is denoted by a subscript within the brackets  $\mathbf{a}_{(i)}$ . We use  $\mathbf{0}$  and  $\mathbf{1}$  to denote vectors of all zeros and all ones, respectively. When we use order relations ( $\geq, \leq, =$ ) between two vectors, it is assumed component-wise. All the vectors are assumed to be column vectors, and  $[a_1, \dots, a_n]^T$  represents a column vector obtained by stacking  $n$  scalars. Matrices are written in bold capital letters  $\mathbf{A}, \mathbf{B}, \mathbf{C}$  and also indexed using subscripts —  $\mathbf{A}_i$  is the  $i^{th}$  row of a matrix  $\mathbf{A}$ , and  $A_{ij}$  is an element of a matrix  $\mathbf{A}$  in a row  $i$  and a column  $j$ . If a superscript is given within the brackets  $\mathbf{A}^{(i)}$  it denotes a specific matrix corresponding to the (vertex) position  $i$ . A notation  $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_n]$  means that a matrix  $\mathbf{A}$  is obtained by stacking vectors  $\mathbf{a}_1, \dots, \mathbf{a}_n$  along the columns. Functions are given using lowercase Latin or Greek characters, but always with corresponding parameters in the brackets  $f(\cdot), g(\cdot), \phi(\cdot), \psi(\cdot)$ . A set of real numbers and a set of positive integers are given by  $\mathbb{R}$  and  $\mathbb{N}$ , respectively. The Euclidean norm is denoted by  $\|\cdot\|$ , and the L1 norm by  $\|\cdot\|_1$ .

## 2 Related Work

Blendshape animation has been a research topic for more than two decades Pighin et al. [1998], Choe and Ko [2006], Choe et al. [2001]. The main tasks in terms of intensive manual labor and extension of production time are (1) the creation of the blendshapes, and (2) the actual animation of the blendshape basis (our work). The main components of a blendshape model are a neutral face mesh and a blendshape basis (local deformations of a neutral face), and there is a body of work proposing automated solutions for creating blendshapes. Two main approaches are (1) building a basis from a dense set of captured data Smet and Gool [2010], Neumann et al. [2013], Chaudhuri et al. [2020], Mengjiao et al. [2020] or (2) deforming a generic set of blendshapes to produce personalized blendshape meshes Li et al. [2010, 2013], Ribera et al. [2017], Zhang et al. [2020], Han et al. [2021], Seol et al. [2016]. In this paper, we assume that the blendshapes are already available and that they closely resemble the actor or user.

The main focus of our paper is to solve the inverse rig problem to produce animation, i.e., to automatically adjust the activation weights so that the resulting mesh follows a reference motion. Reference motion is a (sparse or dense) set of markers recorded from an actor’s face using motion capture (MoCap) systems Deng et al. [2006], Seol and Lewis [2014]. A sparse set of markers is a common approach, particularly if the motion should be retargeted to a fantasy character with a face significantly different from the source actor Sifakis et al. [2005], Kei and Tomoyuki [2012], Seol et al. [2012], Ouzounis et al. [2017], Serra et al. [2018], and it demands special care in positioning the markers on both source and target faces Reverdy et al. [2015]. Although this technique is sufficient for general-purpose MoCap, it fails to capture fine details of the face Furukawa and Ponce [2009], Reverdy et al. [2015]. For this reason, markerless methods are developed to provide high-fidelity performance capture Bradley et al. [2010], Beeler et al. [2011], Thies et al. [2015].

The approaches to solving the inverse rig problem can be divided into data-based (regression models that demand long animated sequences for the training phase) and model-based (that do not demand animation for training, only the rig function with the basis vectors). Data-based solutions are popular due to their ability to provide accurate solutions even for complex rig functions, and commonly apply neural networks Holden et al. [2016], Bailey et al. [2020], Song et al.

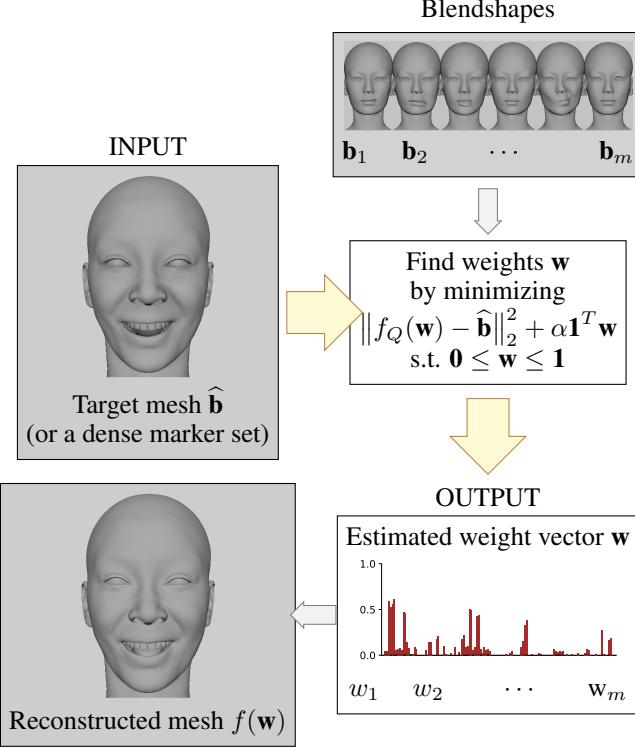


Figure 3: A schematic representation of the considered problem. Some reference mesh  $\hat{\mathbf{b}}$  is taken as an input, and the algorithm needs to estimate the activation weights of the quadratic blendshape model of the character in order to closely reconstruct the reference mesh and respect the structure of the model. The optimization problem might be solved using general-purpose quadratic solvers, like Byrd et al. [1999], or using the algorithm proposed in this paper. Further, the estimated weights are plugged-in into the animation software to give the final reconstructed expression.

[2020], Seonghyeon et al. [2021], radial basis functions Deng et al. [2006], Song et al. [2011], Seol and Lewis [2014], Holden et al. [2015] or other forms of regression Feng et al. [2008], Yu and Liu [2014], Bouaziz et al. [2013]. However, the data acquisition may be too expensive, which is why we consider a model-based approach in this paper. Within model-based approaches, the literature examines only simplified linear blendshape models to fit the acquired mesh, yielding convex optimization problems easy to solve with a closed form Choe and Ko [2006], Sifakis et al. [2005], Çetinaslan [2016], Li et al. [2010], Seol et al. [2011]. In contrast, we specifically target realistic facial animation with a high level of detail, and for this reason, we need to go beyond the linear model and include quadratic corrective terms, as studied in Holden et al. [2016], Song et al. [2017], Kim and Singh [2021].

In this paper, we are not concerned with real-time execution but aim for a more precise mesh fit, hence we assume models with a large number of vertices (we will consider 4000 vertices for each animated character in our experiments). A different line of work addressing the problem of rig inversion is a direct manipulation of the mesh. It aims for the algorithms and tools that are appended to a sculpted character, allowing an artist to refine the pose by dragging specific vertices of the face directly and producing the desired expression Zhang et al. [2004], Lewis and Anjyo [2010], Seo et al. [2011], Cetinaslan and Orvalho [2020a,b]. In this case, it is important that the animator receives feedback during the dragging, hence the optimization takes into account only a sparse set of markers or even a single vertex Lewis and Anjyo [2010], Anjyo et al. [2012].

Despite our formulation being parallelizable, we do not focus on distributed models here, and all the experiments in this paper are performed sequentially over a non-segmented face mesh. Nevertheless, our method is parallelizable, yielding a problem that is separable by components, hence the computations might be distributed to the level of blendshapes. Another approach toward distributed inverse rig solvers is via using face segmentation or clustering. It allows different face regions to be observed and processed independently or in parallel. Early works consider a simple split of the face into upper and lower sets of markers Choe and Ko [2006]. More recent papers model complex splits, either manually Seol et al. [2011], Liu et al. [2010], semi-automatically Na and Jung [2011], Tena et al. [2011], Marco et al. [2020] or automatically James and Twigg [2005], Joshi et al. [2006], Kei and Tomoyuki [2012], Reverdy et al. [2015], Song et al.

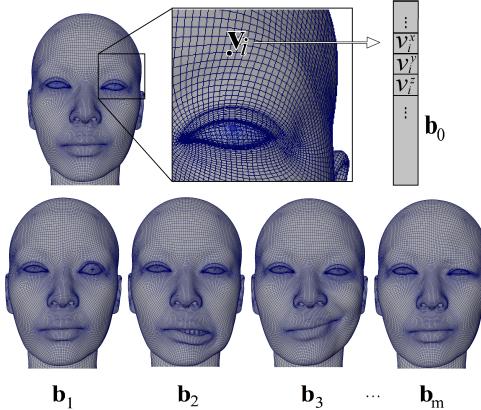


Figure 4: Vectorization of meshes. Neutral mesh  $\mathbf{b}_0$  on top, and example blendshapes below. Each face vertex  $\mathbf{v}_i$  for  $i = 1, \dots, n$  is unraveled into a vector of coordinates  $x, y, z$ , and those coordinate vectors are stacked into a single blendshape vector.

[2017], Bailey et al. [2020]. Clustering based on the underlying deformation model has been considered in Romeo and Schwartzman [2020] and Racković et al. [2021], where the goal of the former was to add a secondary motion to an animated character, and the latter proposes a segmentation for solving the inverse rig locally in a distributed fashion.

All the methods available in the literature solve the inverse rig problem using a linear blendshape function. It turns out it is easy and fast to work with. However, it is of significant interest to work with more complex face models that closely resemble a source actor — a linear model does not exhibit high-enough accuracy for this purpose. In this paper, we introduce a method that allows the application of a more complex blendshape function, that includes quadratic corrective terms, to produce an accurate and sparse solution to the inverse rig problem.

Further details on the principles of blendshape animation can be found in references Lewis et al. [2014] and Çetinaslan [2016].

### 3 Background on Rig Approximation and Inverse Rig

In this section, we give a concise presentation on the main principles of blendshape animation. Section 3.1 introduces the linear delta blendshape model. Section 3.2 introduces quadratic corrective terms that are added on top of a linear model in order to increase the mesh fidelity of realistic human characters. Finally, Section 3.3 explains how inverse rig problems have been formulated and solved according to existing literature.

#### 3.1 Linear Blendshape Model

Traditionally, a *blendshape model* consists of a neutral face, that is represented by a column vector  $\mathbf{b}_0 \in \mathbb{R}^{3n}$ , and a set of  $m$  blendshape vectors  $\mathbf{b}_1, \dots, \mathbf{b}_m \in \mathbb{R}^{3n}$  that correspond to atomic expressions obtained by local deformations over  $\mathbf{b}_0$  (Figure 4). Each blendshape  $\mathbf{b}_i$  is assigned an activation parameter  $w_i$  that usually (but not exclusively) takes values between 0 and 1 Seol et al. [2012]. It is common to use a delta formulation of the blendshape model, where the elements of the blendshape vectors  $\mathbf{b}_1, \dots, \mathbf{b}_m$  are not the actual coordinates of the deformed face, but the offsets from their corresponding positions in the neutral  $\mathbf{b}_0$ . (For this reason, some authors use the notation  $\Delta\mathbf{b}_i$  instead of  $\mathbf{b}_i$ ; however, we will not use the  $\Delta$  symbol in order to ease the notation.) A *linear delta blendshape function*  $f_L(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^{3n}$  maps activation weights  $w_1, \dots, w_m$  onto the mesh space, and it is defined as

$$f_L(w_1, \dots, w_m) = \mathbf{b}_0 + \sum_{i=1}^m w_i \mathbf{b}_i. \quad (2)$$

If we collect blendshape vectors into a matrix  $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_m]$ ,  $\mathbf{B} \in \mathbb{R}^{3n \times m}$ , the function can be written in a matrix form as

$$f_L(\mathbf{w}) = \mathbf{b}_0 + \mathbf{B}\mathbf{w}, \quad (3)$$

where  $\mathbf{w} = [w_1, \dots, w_m]^T$  represents a column vector of blendshape weights.

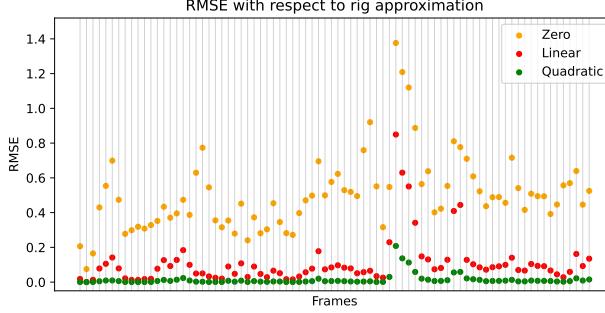


Figure 5: RMSE between ground truth meshes and meshes obtained using different approximations, for the animated character *Omar*. Besides the linear and quadratic blendshape model, an additional *zero* approximation is included to give a better idea of the error scale — it corresponds to a difference between the original mesh and a neutral expression.

### 3.2 Quadratic Blendshape Model

In modern animation, with an increasing level of detail and with avatars that closely resemble an actor (or a user), linear models are too restrictive and fail to span a desired space of motion. For this reason, additional *corrective shapes* (also known as combination shapes) are included Seo et al. [2011], Lewis et al. [2014], and these are usually more numerous than the base vectors. In particular, the quadratic corrective terms are very common, and adding them on top of a linear function (3) significantly improves the accuracy of the representation; hence, we introduce a quadratic blendshape model in the following lines.

A pair of blendshapes  $\mathbf{b}_i$  and  $\mathbf{b}_j$  that deform the same local area can produce mesh artifacts when activated simultaneously, so the additional corrective term  $\mathbf{b}^{\{i,j\}} \in \mathbb{R}^{3n}$  is included to adjust the resulting deformation. It is constructed as  $\mathbf{b}^{\{i,j\}} = \hat{\mathbf{b}}^{\{i,j\}} - (\mathbf{b}_0 + \mathbf{b}_i + \mathbf{b}_j)$ , where  $\hat{\mathbf{b}}^{\{i,j\}}$  represents a desired result of joint activation of deformers  $i$  and  $j$  (an artist sculpts it manually). Now, whenever the blendshapes  $\mathbf{b}_i$  and  $\mathbf{b}_j$  are activated simultaneously, the corrective blendshape  $\mathbf{b}^{\{i,j\}}$  is activated as well, so that the corrective contribution due to simultaneous activation of  $\mathbf{b}_i$  and  $\mathbf{b}_j$  equals  $w_i w_j \mathbf{b}^{\{i,j\}}$ . A *quadratic blendshape function*  $f_Q(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^{3n}$  can now be defined as

$$f_Q(\mathbf{w}) = \mathbf{b}_0 + \mathbf{B}\mathbf{w} + \sum_{(i,j) \in \mathcal{P}} w_i w_j \mathbf{b}^{\{i,j\}}, \quad (4)$$

where  $\mathcal{P}$  represents a set of tuples  $(i, j)$  such that there is a quadratic corrective term between corresponding blendshapes  $\mathbf{b}_i$  and  $\mathbf{b}_j$ . In practice, it is common to use additional levels of corrections, yielding even more complex blendshape models — all the animated characters that we use for the experiments in this paper have at least one more level of corrections. However, the quadratic terms (first level of correction) already give a considerably more accurate approximation of the rig compared to a simple linear model — in Figure 5 we compare the error of reconstruction of the ground truth animation frames, under linear and quadratic blendshape model approximations, for the animated character *Omar* (for a complete description of datasets see Section 5). Regardless of their wide use in practice, the corrective terms of the blendshape model are largely overlooked in the literature.

### 3.3 Inverse Rig Problem

In this section, we describe the concept that is central to this paper — the inverse rig problem for facial animation. Further, we cover state-of-the-art solutions before introducing our novel approach in the next section.

The *inverse rig problem* in automatic motion retargeting is the problem of finding optimal activation parameters  $\mathbf{w}$  (blendshape weights) to produce a target mesh  $\hat{\mathbf{b}}$ , which is usually given as a 3D scan of an actor or a set of MoCap markers. It is common to pose the problem in the least squares framework:

$$\underset{\mathbf{w}}{\text{minimize}} \|f(\mathbf{w}) - \hat{\mathbf{b}}\|^2, \quad (5)$$

where  $f(\mathbf{w}) : \mathbb{R}^m \rightarrow \mathbb{R}^{3n}$  is a rig function,  $\hat{\mathbf{b}} \in \mathbb{R}^{3n}$  is a target mesh, and additional constraints and regularization terms might be included. Regularization terms are added to produce a more stable solution, but might also help to make vector  $\mathbf{w}$  sparser — this is desirable because animators usually need to alter the solutions by hand, which gets much harder if a large number of blendshapes are already activated Seol et al. [2011]. In Joshi et al. [2006], the objective

function takes exactly the form of (5); in Choe and Ko [2006], Liu et al. [2010] the constraints are invoked to keep the weights in the  $[0, 1]$  interval; Lewis and Anjyo [2010], Çetinaslan [2016] add a regularization term  $\|\mathbf{w}\|^2$ , while Ribera et al. [2017] uses a  $\|\mathbf{w}\|_1$  regularization, which is known to enhance sparsity of the solution. Importantly, all the mentioned papers assume the blendshape model is linear.

We take the formulation given in Cetinaslan and Orvalho [2020a], as a state-of-the-art approach using a linear blendshape model:

$$\underset{\mathbf{w}}{\text{minimize}} \|\mathbf{Bw} - \hat{\mathbf{b}}\|^2 + \alpha \|\mathbf{w}\|^2, \quad (6)$$

where  $\alpha \geq 0$ . The weight constraints are not explicitly included in the optimization, but the values of the resulting vector that are outside of the feasible set are clipped afterward in order to satisfy the model constraints. Note that a neutral face  $\mathbf{b}_0$  is omitted in the above formulation, hence the target  $\hat{\mathbf{b}}$  is also taken as an offset from the neutral face and not an actual mesh. One adjustment to the above approach, given in the same paper, is using a sparse approximation  $\mathbf{B}^{loc}$  of a matrix  $\mathbf{B}$ , instead of an actual blendshape matrix. This excludes irrelevant blendshape effects in the local regions and leads to a sparser solution and lower computational cost; however, it might affect the accuracy of the reconstructed mesh.

A different approach is given by Seol et al. [2011], where the problem is solved sequentially, for a single blendshape at a time (*step 1* below), and the residual mesh  $\hat{\mathbf{b}}$  is updated after each iteration (*step 2*), before proceeding for the next controller:

$$\begin{aligned} \text{step 1: } w_i &= \underset{w}{\text{argmin}} \|\mathbf{b}_i w - \hat{\mathbf{b}}\|^2 \\ \text{step 2: } \hat{\mathbf{b}} &\leftarrow \hat{\mathbf{b}} - \mathbf{b}_i w_i. \end{aligned} \quad (7)$$

The order in which weights are optimized is crucial here. The authors suggest sorting them according to the average magnitude of deformation each blendshape produces over a whole face. This method yields a sparse solution and avoids simultaneous activation of mutually exclusive blendshapes Seol et al. [2011], Lewis et al. [2014].

To the best of our knowledge, no prior work considered adding the non-linear terms in the objective when solving the inverse rig problem. While working with linear approximation yields a simple and fast solution, it lacks accuracy and fails to closely resemble a source actor. In the next section, we introduce a method that allows the application of the quadratic blendshape function (4) to produce an accurate and sparse solution for the inverse rig problem.

## 4 Proposed Method

A linear blendshape function is, in general, convenient for solving the inverse rig problems, especially for cartoonish characters whose face proportions differ significantly from a source actor. However, we propose a more suitable model for a specific subdomain of facial animation — character face models that are sculpted to closely resemble a source actor, which demands higher accuracy compared to what a linear function offers. We consider the quadratic blendshape function (4). Additionally, we assume that activation weights  $w_1, \dots, w_m$  must stay within the  $[0, 1]$  interval<sup>1</sup>.

We will first formulate the objective function that should be minimized in order to provide the desired solution and show how it can be solved using standard ad-hoc quadratic solvers. However, we will show experimentally that such a solution is not entirely satisfactory; hence we further introduce an algorithm that we developed for solving this specific optimization problem, which yields a solution with an optimal trade-off between several metrics important for the domain application.

### 4.1 Objective Function

Our objective will encode the two most desired properties of the solution to the inverse rig problem: (1) the difference between a reference and reconstructed mesh should be as low as possible, hence we want to incorporate the quadratic corrective terms instead of using only a linear approximation of the blendshape model; (2) the solution vector should exhibit low cardinality, i.e., the number of non-zero elements, since that helps both for the posterior manual modification, and ensuring the stability of the solution; and finally, the weights must stay in the feasible interval  $[0, 1]$ . The proposed objective takes into account all the above assumptions:

$$\underset{0 \leq w \leq 1}{\text{minimize}} \|f_Q(\mathbf{w}) - \hat{\mathbf{b}}\|^2 + \alpha \mathbf{1}^T \mathbf{w}. \quad (8)$$

<sup>1</sup>Some authors allow for values outside this interval, either for the sake of simplicity or because it might be beneficial for cartoon characters. However, in the subdomain of animation targeted by our work, the construction of the animation models does not permit violating this constraint.

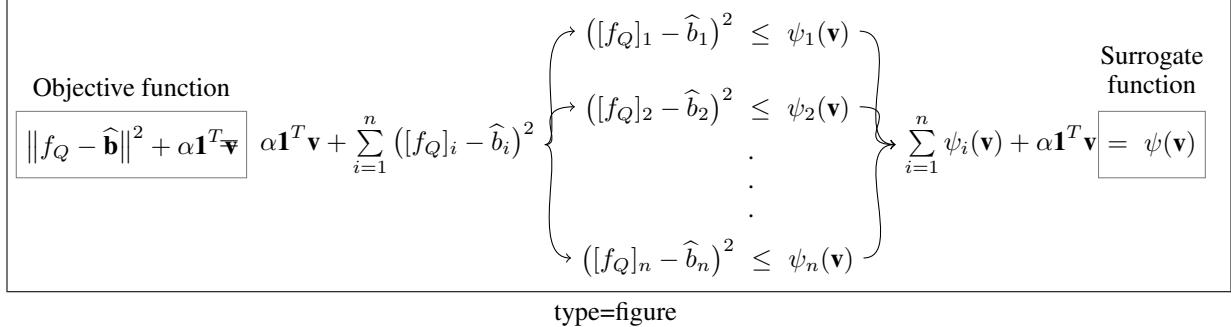


Figure 6: Graphical representation of the construction of the proposed upper bound function  $\psi(\cdot)$ . To simplify the representation, in this figure we use a notion  $f_Q$  instead of  $f_Q(\mathbf{w} + \mathbf{v})$ .

The regularization term in (8), with  $\alpha \geq 0$ , is invoked to encourage a sparse solution — note that, as  $\mathbf{w} \geq \mathbf{0}$  in the feasible set,  $\mathbf{1}^T \mathbf{w}$  equals the  $L1$  norm of  $\mathbf{w}$ , which is known to be a sparsity-enhancing regularizer Schmidt [2005].

We show that quadratic terms increase quality by using standard solvers and increase performance with a dedicated algorithm. Compared to the approaches described in Section 3.3, this formulation is a hard-to-solve nonconvex problem and, in particular, does not allow for a closed-form solution. However, it can be readily solved using some standard solvers. Specifically, in this paper, we show the results of applying the method from Byrd et al. [1999] that is a barrier (interior point) method in which the subproblems are solved approximately by a sequential quadratic programming (SQP) iteration with trust regions Conn et al. [2000], Wright et al. [1999]. We decided to use Byrd et al. [1999] as it corresponds to a common python library implementation of a general-purpose solver that is suitable for our problem – it handles constrained, nonconvex problems – while utilizing SQP and interior point ideas<sup>2</sup>. This approach yields a high-accuracy fit of the solution, thus demonstrating the advantage of using quadratic terms.

However, this is a general-purpose algorithm, and as expected, the solution obtained by it exhibits other issues. We will show in our experiments that, even though the data fidelity is exceptionally high, the resulting weight vector activates all the elements, yielding an undesirable solution. On top of that, frame-to-frame transitions are far from being smooth. To address all these issues, we propose a novel algorithm for solving the problem (8).

## 4.2 Proposed Algorithm

The objective in (8) is hard to minimize, hence we resort to the *majorization-minimization* Schifano et al. [2010], Sun et al. [2016] technique to obtain a good-enough approximate solution that clearly improves on the linear model. The idea is to approximate the objective with an iteration over functions  $\psi(\cdot)$ , which are easier to solve. In the following lines, we will explain how the appropriate function is constructed (Figure 6). Further, we will see that it is separable by coordinates, hence we estimate each blendshape weight independently, solving the set of simple problems. This approach is iterative and suitable for parallelization since the weights are fitted independently. Further, as we will confirm in the numerical experiments, the obtained solution is accurate while keeping the low cardinality (the number of non-zero weights) and smooth transitions over the frames.

We start by rewriting the objective (8) in a *Levenberg-Marquardt* fashion Ananth [2004], i.e., we assume some initial weight vector  $\mathbf{w}$  is available, and we are looking for an increment vector  $\mathbf{v}$  that would lead to a better solution  $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{v}$ . Optimization problem is then reformulated as

$$\underset{-\mathbf{w} \leq \mathbf{v} \leq \mathbf{1}-\mathbf{w}}{\text{minimize}} \|\mathbf{f}_Q(\mathbf{w}+\mathbf{v}) - \hat{\mathbf{b}}\|^2 + \alpha \mathbf{1}^T \mathbf{v}. \quad (9)$$

A quadratic term in function  $f_Q$  makes this problem too complex to solve exactly, so we apply a majorization-minimization paradigm that solves an approximate problem. We introduce an upper bound function  $\psi(\mathbf{v}; \mathbf{w}) : \mathbb{R}^m \rightarrow \mathbb{R}$  over the above objective (9), such that  $\|\mathbf{f}_Q(\mathbf{w}+\mathbf{v}) - \hat{\mathbf{b}}\|^2 + \alpha \mathbf{1}^T \mathbf{v} \leq \psi(\mathbf{v}; \mathbf{w})$  holds for  $\mathbf{0} \leq \mathbf{w} + \mathbf{v} \leq \mathbf{1}$ . This bound is easier to minimize than the original objective, and we continue with the problem in the form

$$\underset{-\mathbf{w} \leq \mathbf{v} \leq \mathbf{1}-\mathbf{w}}{\text{minimize}} \psi(\mathbf{v}; \mathbf{w}). \quad (10)$$

<sup>2</sup>[docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html](https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html)  
[docs.scipy.org/doc/scipy/reference/optimize.minimize-trustconstr.html](https://docs.scipy.org/doc/scipy/reference/optimize.minimize-trustconstr.html)

In other words, for a current solution estimate  $\mathbf{w}$ , we search for increment  $\mathbf{v}$  to construct the new solution estimate  $\mathbf{w}+\mathbf{v}$  by minimizing a surrogate function  $\psi(\mathbf{v}; \mathbf{w})$  in (10) instead of the original cost function in (9). The surrogate function (a global upper bound on the cost function in (9)) is carefully constructed such that it represents a good,  $\mathbf{w}$ -dependent approximation of the cost in (9) around the current point  $\mathbf{w}$ , and such that (10) is easy to solve. Further, it needs to respect the following two conditions:

- for any feasible vector  $\mathbf{0} \leq \mathbf{w} \leq \mathbf{1}$ , and any increment vector  $\mathbf{v}$  such that  $\mathbf{0} \leq \mathbf{w} + \mathbf{v} \leq \mathbf{1}$ , the original objective is bounded by the surrogate from above, i.e.,

$$\|f_Q(\mathbf{w}+\mathbf{v}) - \hat{\mathbf{b}}\|^2 + \alpha \mathbf{1}^T \mathbf{v} \leq \psi(\mathbf{v}; \mathbf{w}); \quad (11)$$

- at point  $\mathbf{v} = \mathbf{0}$ , the values of the objective and of the majorizer function are equal, that is,

$$\|f_Q(\mathbf{w}) - \hat{\mathbf{b}}\|^2 = \psi(\mathbf{0}; \mathbf{w}). \quad (12)$$

In the rest of this paper we will write  $\psi(\mathbf{v})$  instead of  $\psi(\mathbf{v}; \mathbf{w})$  for the sake of simplicity. We will consider a surrogate function such that the regularization term in (9) is kept the same, and we only approximate the data fidelity term. Note that the data fidelity term is a sum of squares over the mesh coordinates. Hence, we will look for the bound that can also be written as a sum, and bound the function at each of the coordinates:

$$\psi(\mathbf{v}) = \sum_{i=1}^{3n} \psi_i(\mathbf{v}) + \alpha \mathbf{1}^T \mathbf{v}, \quad (13)$$

where  $\psi_i(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}$  is constructed to upper-bound the data fidelity term in the mesh coordinate  $i$ , i.e.,  $([f_Q(\mathbf{w}+\mathbf{v})]_i - \hat{b}_i)^2 \leq \psi_i(\mathbf{v})$ . The notation  $[f_Q(\mathbf{w}+\mathbf{v})]_i$  indicates that we are observing the  $i^{th}$  coordinate of the resulting mesh. We want to rewrite this in a canonical quadratic matrix form, hence we introduce a symmetric matrix  $\mathbf{D}^{(i)} \in \mathbb{R}^{m \times m}$ , such that its elements are coordinates of the corrective terms for the corresponding blendshape pairs,  $D_{jk}^{(i)} = \frac{1}{2}b_i^{\{j,k\}}$ . Now we have  $[f_Q(\mathbf{w})]_i = \mathbf{B}_i^T \mathbf{w} + \mathbf{w}^T \mathbf{D}^{(i)} \mathbf{w}$ , so the coordinate-wise surrogate function should satisfy

$$\left( \mathbf{B}_i^T (\mathbf{w} + \mathbf{v}) + (\mathbf{w} + \mathbf{v})^T \mathbf{D}^{(i)} (\mathbf{w} + \mathbf{v}) - \hat{b}_i \right)^2 \leq \psi_i(\mathbf{v}). \quad (14)$$

Since we are minimizing with respect to the increment vector  $\mathbf{v}$ , the vector  $\mathbf{w}$  is a constant; hence the left-hand side can be simplified to

$$(g_i + \mathbf{h}_i^T \mathbf{v} + \mathbf{v}^T \mathbf{D}^{(i)} \mathbf{v})^2, \quad (15)$$

where  $g_i = \mathbf{B}_i^T \mathbf{w} + \mathbf{w}^T \mathbf{D}^{(i)} \mathbf{w} - \hat{b}_i$ , and  $\mathbf{h}_i = \mathbf{B}_i + \mathbf{w}^T \mathbf{D}^{(i)}$  are the terms that do not depend on  $\mathbf{v}$ . The upper bound function is then derived by bounding the non-linear terms of the above expression separately (a linear term is kept the same). A coordinate-wise bound that satisfies the above conditions is

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

where a function  $\lambda_M(\cdot, \cdot) : (\mathbb{R}^{m \times m}) \rightarrow \mathbb{R}$  is defined as

$$\lambda_M(\mathbf{D}^{(i)}, g_i) := \begin{cases} \lambda_{\min}(\mathbf{D}^{(i)}), & \text{if } g_i < 0, \\ \lambda_{\max}(\mathbf{D}^{(i)}), & \text{if } g_i \geq 0, \end{cases} \quad (17)$$

and  $\lambda_{\min}(\mathbf{D}^{(i)})$ ,  $\lambda_{\max}(\mathbf{D}^{(i)})$  stand for the minimum and maximum eigenvalues of the matrix  $\mathbf{D}^{(i)}$  respectively, and  $\sigma(\mathbf{D}^{(i)})$  is the largest singular value of the matrix  $\mathbf{D}^{(i)}$ .

A complete bound is reached by summing all coordinate-wise bounds and adding a regularization term:

$$\psi(\mathbf{v}) = \sum_{i=1}^n \left( g_i^2 + 2g_i \sum_{j=1}^m h_{ij} v_j + 2 \left( g_i \lambda_M(\mathbf{D}^{(i)}, g_i) + \|\mathbf{h}_i\|^2 \right) \sum_{j=1}^m v_j^2 + 2m\sigma^2(\mathbf{D}^{(i)}) \sum_{j=1}^m v_j^4 \right) + \alpha \mathbf{1}^T \mathbf{v}. \quad (18)$$

Figure 6 illustrates the above process.

One can see that the upper bound (18) is separable by components, and that the solution will be available in closed form. If we consider a single blendshape index  $j \in \{1, \dots, m\}$  and regroup the coefficients of the bound function as

$$q = \sum_{i=1}^{3n} g_i h_{ij} + \alpha, \quad r = \sum_{i=1}^{3n} (g_i \lambda_M(\mathbf{D}^{(i)}, g_i) + \|\mathbf{h}_i\|^2), \quad s = m \sum_{i=1}^{3n} \sigma^2(\mathbf{D}^{(i)}), \quad (19)$$

we can write an objective function in the form of a quartic equation without a cubic term:

$$\begin{aligned} & \underset{v_j}{\text{minimize}} \quad q v_j + r v_j^2 + s v_j^4, \\ & \text{s.t. } 0 \leq w_j + v_j \leq 1. \end{aligned} \quad (20)$$

This procedure is summarized in Algorithm 1 and we refer to it as an *inner iteration*. Algorithm 1 solves the problem (20) for each component to give a full increment vector  $\mathbf{v}$ , that is, solving (18) is equivalent to solving (20) for each  $j = 1, \dots, m$  independently. In this sense, our approach is somewhat similar to the solution of Seol et al. [2011], as given in (7); however, we do not update vector  $\mathbf{w}$  before all the components are optimized, which helps to avoid the issues with making the right update order, and additionally opens the possibility for a parallel implementation of the procedure.

---

**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  $i = 1, \dots, 3n$ , target mesh  $\hat{\mathbf{b}} \in \mathbb{R}^{3n}$ , regularization parameter  $\alpha > 0$  and weight vector  $\mathbf{w} \in [0, 1]^m$ .

**Ensure:** An optimal increment vector  $\hat{\mathbf{v}}$  as a solution to (20).

Compute coefficients  $q, r$  and  $s$  from eq. (19) 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**  $j = 1, \dots, m$  **do**

$$q = 2 \sum_{i=1}^{3n} g_i h_{ij} + \alpha$$

$$\hat{v}_j = \underset{v}{\text{argmin}} \quad q v + r v^2 + s v^4$$

$$\text{s.t. } -w_j \leq v \leq 1 - w_j$$

**end for**

**return**  $\hat{\mathbf{v}}$

---

The solution of the inner iteration will depend on the initial weight vector, hence we repeat the procedure in Algorithm 1 multiple times in order to provide an increasingly good estimate  $\mathbf{w}$  of the solution to (9), as explained in Algorithm 2. After each iteration  $t = 1, \dots, T$ , we update the weight vector as  $\mathbf{w}_{(t+1)} = \mathbf{w}_{(t)} + \mathbf{v}_{(t)}$ . An initial vector can be chosen anywhere within the feasible space  $\mathbf{0} \leq \mathbf{w}_{(0)} \leq \mathbf{1}$ , but in Section 5 we mention strategies for initialization based on domain knowledge, that lead to faster convergence and yield better results.

Additional details on the construction of the surrogate and a discussion of the algorithm convergence are outside of the scope of this work, but we include them in the companion paper not for review [2022].

## 5 Evaluation

As mentioned earlier, we consider realistic human characters with a high level of detail and activation weights restricted to lie between 0 and 1. The first three characters that we present in these results are publicly available within the MetaHuman<sup>3</sup> platform — *Omar*, *Danielle* and *Myles* ([unrealengine.com/en-US/eula/mhc](https://unrealengine.com/en-US/eula/mhc)), as shown in Figure 7. The additional two datasets that we used to evaluate the method are the property of the animation studio — *Char 4* and *Char 5*. All the characters are accompanied by a short animation sequence covering a wide range of facial expressions that were used to evaluate the methods<sup>4</sup>. We exclude inactive vertices and the vertices in the neck and shoulder regions for each character, so after the subsampling, each model has  $n = 4000$  vertices. The scale of the head is also similar between all the characters, and the width between the left and right ear is approximately 18 cm. However, the number of blendshapes differs (ranging between 60 and 150), and that means that different choices of the regularization parameter

<sup>3</sup>We chose MetaHuman characters since they align well with the assumptions of our model and are an increasingly popular choice for building realistic face animation Fang et al. [2021], Siddiqui [2022] ([unrealengine.com/en-US/metahuman](https://unrealengine.com/en-US/metahuman))

<sup>4</sup>Upon the acceptance of the paper, the animation weights for *Omar*, *Danielle* and *Myles* will be available in a public repository, together with the code.

**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  $\alpha > 0$ , initial weight vector  $\mathbf{w}_{(0)} \in [0, 1]^m$ , maximum number of iterations  $T \in \mathbb{N}$  and the tolerance  $\epsilon > 0$ .

**Ensure:**  $\widehat{\mathbf{w}}$  - an approximate minimizer of the problem (8).

For each vertex  $i$  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  $i = 1, \dots, 3n$  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
for  $t = 0, \dots, T$  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
         $\widehat{\mathbf{w}} \leftarrow \mathbf{w}_{(t+1)}$ 
        return  $\widehat{\mathbf{w}}$ 
    end if
end for
 $\widehat{\mathbf{w}} \leftarrow \mathbf{w}_{(t+1)}$ 
return  $\widehat{\mathbf{w}}$ 

```

---

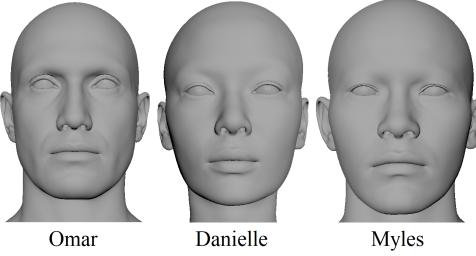


Figure 7: Head models available at MetaHuman Creator platform (©unrealengine.com).

$\alpha \geq 0$  (Eq. (6) and (8)) might be optimal for various models. It is important to note that *Char 5* has a more complex rig than the other four characters, with a number of deformers that are not based on a blendshape deformation (rotational and joint-like deformers); still, we include it in the experiments to show that our algorithm is robust enough to produce satisfying results even in such a case.

### 5.1 Benchmark Methods

As mentioned in Section 3.3, a state-of-the-art representative of model-based approaches to solving the inverse rig problem is a method given by Cetinaslan and Orvalho [2020a] in Eq. (6) — in the rest of this paper we will denote this approach as *Cet*.

In the same paper Cetinaslan and Orvalho [2020a], the authors propose a modification of the solution using the heat kernel of Crane et al. [2013] to transform an original blendshape matrix  $\mathbf{B}$  into a sparse approximation  $\mathbf{B}^{loc}$ . The idea is that the vertices of the face should not affect the activation of blendshapes whose main impact is localized in a distant face region. Hence, this method provides localized and more stable results. The optimization problem for that case is posed identically to (6), except that matrix  $\mathbf{B}$  is substituted with  $\mathbf{B}^{loc}$ . This approach will be denoted by *Cet-loc*.

A different approach is proposed in Seol et al. [2011], where the weights are optimized sequentially (7), starting from the ones that have a larger overall effect on the face. This method does not include any weight constraints, yet our animated characters are strictly demanded to have values between 0 and 1. Hence, when applying this method, we

project the estimated weight onto a feasible interval after each iteration. This method will be denoted as *Seol*. Note that in this approach, there is no regularization parameter.

Our method will be evaluated for two cases, as mentioned earlier. The first approach is to consider the proposed objective function (8), but minimize it by applying a general-purpose quadratic solver — we use the `scipy` implementation of the interior point solver Byrd et al. [1999], whose inner iterations are solved by applying sequential quadratic programming technique and trust region method<sup>5</sup>. We denote this *SQP* solution. The second approach solves the problem (8) using the majorization-minimization-based algorithm proposed in Section 4.2, and it will be denoted *MM*. The solutions might be affected by choice of the initial point. In Section 4, we mentioned that the algorithm might be initialized with any feasible  $\mathbf{w}$ ; however, random initialization leads to slower convergence, and the results are often poor in terms of both mesh fidelity and sparsity, hence we aim for a more educated guess of the initial vector. A simple choice is  $\mathbf{w} = \mathbf{0}$ , as this would lead to a sparse solution. The results of our method obtained with zero initialization will be denoted *MM-0*. Another possibility is initializing our method with the solution of a problem under the linear blendshape function approximation. In this case, we chose a solution of *Cet* because it gives a solution in closed form using a pseudoinverse of a matrix  $\mathbf{B}$ ; hence, we refer to this as *MM-psd*. In the same way, we can use the results of *Seol* to initialize our method, however, it does not show any advantage over the other, hence we proceed with the above two initialization strategies. (*SQP* is also an iterative solver, but in our experiments different initialization strategies did not show significant differences in the results. For this approach, all the presented results are initialized with a zero vector.)

## 5.2 Metrics

In order to evaluate the goodness of the results, we will consider several different metrics. The main metric of interest is a root mean squared error (RMSE), which will serve to measure the data fidelity of the results and is defined as:

$$RMSE(\hat{\mathbf{b}}, \tilde{\mathbf{b}}) = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{b}_i - \tilde{b}_i)^2}, \quad (21)$$

where  $\hat{\mathbf{b}}$  is a target face mesh and  $\tilde{\mathbf{b}}$  is a predicted mesh. The meshes  $\tilde{\mathbf{b}}$  are obtained by plugging the predicted weight vectors in Autodesk Maya Autodesk, INC. [2019-01-15], hence all the methods are evaluated on the same level of rig complexity. One important thing to note here is that RMSE in this standard form might not be the best choice for assessing the results of mesh reconstruction. The problem is that, when a large number of vertices is only slightly misplaced, the value of the metric might be similar to the case when only a small group of vertices is misplaced but to a large extent. We need to rate these two scenarios differently since the latter is what actually produces visible misfits. In order to accomplish this, we will not use the mean value of the squared distances as in (21), but instead, we take the 95<sup>th</sup> percentile of the error.

While the mesh error is the metric of primary importance to us, we also want a solution to be as sparse as possible while keeping the mesh fidelity high (i.e., RMSE low). An appropriate metric for this is the cardinality of a predicted weight vector, i.e., the number of non-zero elements of  $\mathbf{w}$ . Some estimated weights might have values that are very close to zero and, in practice, negligible, but they will still count when measuring cardinality. For this reason, we include an additional indicator of the sparsity —  $L_1$  norm of the solution vector.

Finally, as mentioned earlier, we would prefer if the weights for the consecutive frames of the animation sequence have smooth transitions. If we take a single blendshape weight  $w_i$  and observe its values over  $T$  time frames, we get a discrete time series  $[w_i^{(1)}, \dots, w_i^{(T)}]$ . To evaluate its smoothness, we will use the second-order differences Green and Silverman [1993], Mehlm and Tarrou [1998]

$$\text{Smoothness factor}(w_i) = \sum_{t=2}^{T-1} (w_i^{(t-1)} - 2w_i^{(t)} + w_i^{(t+1)})^2. \quad (22)$$

Lower values indicate smoother time series, while the minimum value of 0 is achieved for a constant vector. Note also that this metric is evaluated per component over time, while the previous ones take a static frame and evaluate over the entire set of weights/vertices.

## 5.3 Numerical Results

The experiments are executed on a user-level computer with the processor Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz and 8GB of RAM.

---

<sup>5</sup>[docs.scipy.org/doc/scipy/reference/optimize.minimize-trustconstr.html](https://docs.scipy.org/doc/scipy/reference/optimize.minimize-trustconstr.html)  
[docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html](https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html)

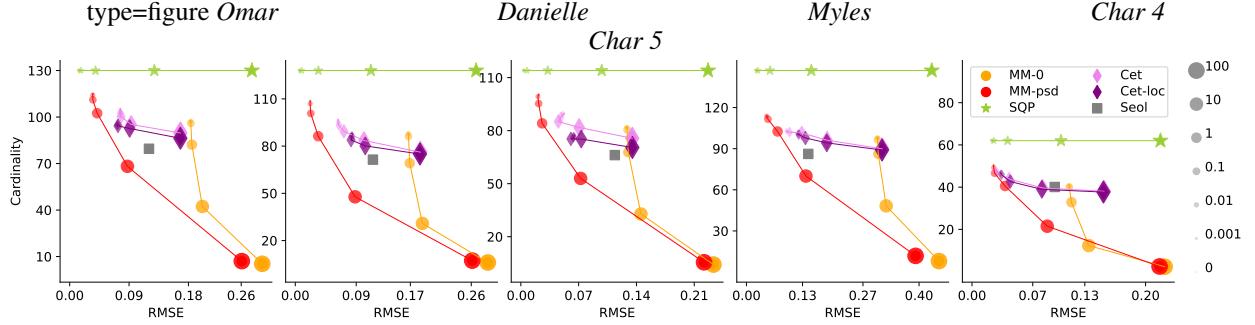


Figure 8: A trade-off between mesh error (RMSE) and cardinality (i.e. the number of non-zero weights) of the estimated solutions for different methods and varying values of regularization parameter  $\alpha$ . Different marker sizes correspond to values of  $\alpha$ , as indicated on the far right.

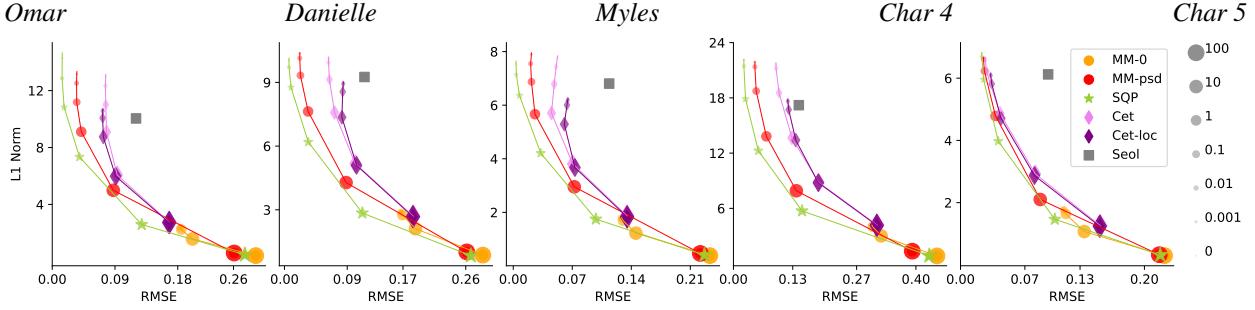


Figure 9: A trade-off between mesh error (RMSE) and L1 norm of the estimated solutions for different methods and varying values of regularization parameter  $\alpha$ . Different marker sizes correspond to values of  $\alpha$ , as indicated on the far right.

Our method, under *SQP* and under *MM*, as well as the other benchmark models, are tested with a wide range of regularization parameter values  $\alpha \in \{0, 0.001, 0.01, 0.1, 1, 10, 100\}$ . We chose this interval of values as it covers both extremely low and extremely high values, hence we expect the optimal value to be somewhere within. However, a reader should be aware that this range might be considerably different depending on the considered animation character. For example, in Cetinaslan and Orvalho [2020a], the authors mention that the values of  $\alpha > 1$  make the regularization term completely dominates the objective, yet in their experiments with direct manipulation, only a few face vertices are included in the objective. On the other side, we use  $n = 4000$  vertices for each character in our retargeting experiments; hence, the regularization term takes dominion only when the value of  $\alpha$  is much closer to 100. Besides the number of vertices  $n$ , the size of the head model (or the units of measure) and the number of blendshapes  $m$  will also affect the choice of  $\alpha$ .

The desired solution for the rig inversion should have high data fidelity while keeping the number of activated components low, hence we look at the trade-off curves between RMSE (mesh error) and cardinality of the weight vectors in Figure 8. Additionally, Figure 9 complements the results showing a trade-off between RMSE and L1 norm of the weight vectors. Predictions are made over the training sets of 80 animated frames for each character. The training frames are subsampled from the original animation sequence so that there are at least 10 time units between any two frames in the resulting set — this was done in order to avoid the redundancy of the consecutive frames Tena et al. [2011].

The results of *Seol* are presented by a single point (a gray square) in Figures 8 and 9, since the approach of Seol et al. [2011] does not include any regularization. The other methods are presented for different values of  $\alpha$  indicated by the size of a corresponding marker and connected by a same-color line. The first thing to notice is that *SQP* (green stars) always activates all the blendshape weights, yielding a trade-off curve that is equal to  $m$  on the  $y$ -axis. On the other side, if we consider the plot of RMSE versus L1 norm, its trade-off is favorable, and actually, *SQP* curves are always under the other methods. This tells us that, most probably, many of those components are activated by a value that is very close to zero, and if further effort is invested, this method might be adjusted to give low cardinality and

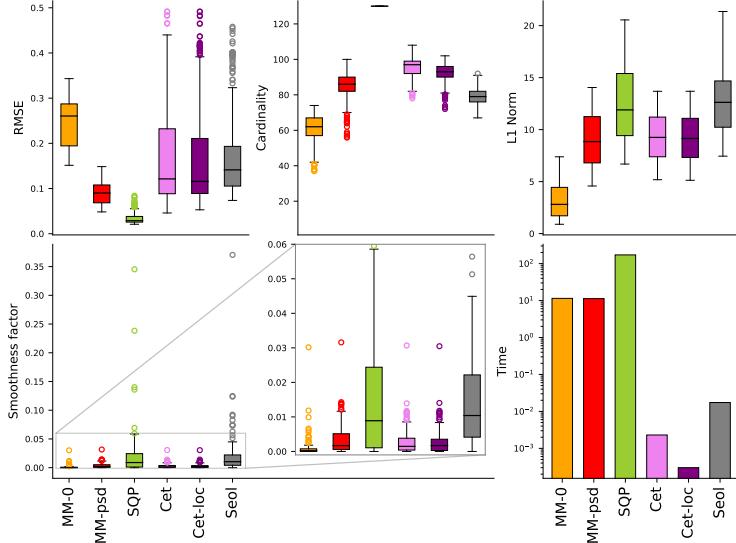


Figure 10: Values of the four metrics (mesh RMSE, weights cardinality, weights  $L_1$  norm, and temporal smoothness of weight curves) and execution time (in seconds) for *Omar*. Execution time is presented in a log-scale, because of the wide range — for *Cet-loc* it takes 0.0003 s, and for *SQP* 172.7 s. See Table 1 for numerical details.

high-accuracy reconstructions. Results of *MM* are presented in round dots (red for *psd* and orange for 0 initialization). While the zero initialization shows a relatively high mesh error, *MM-psd* gives a trade-off curve that is always below those of the other methods, indicating its superiority. Results of *Cet* and *Cet-loc* (light and dark violet diamonds, respectively) are relatively similar, with the localized version having slightly lower cardinality and higher RMSE.

To compare the results in more detail, we pick an optimal value of  $\alpha$ , using the elbow technique, for each approach and each character, and evaluate the predictions over the test set. Test sets consist of 500 frames for each character. In this case, we take the animation sequences instead of the isolated frames. This will allow us to compare the smoothness of the results and to produce the animated clips in order to visually inspect the properties of different methods (check the supplementary video materials).

Figure 10 gives the values of the four metrics for *Omar* over the test set, as well as the average computational time per frame. Boxplots show the median values and quartiles (across 500 frames in case of RMSE, cardinality, and  $L_1$ ; and across  $m = 130$  blendshape weights in the case of smoothness factor), while the mean values are given in Table 1. Table 1 additionally contains the selected optimal  $\alpha$  values (notice that the table contains only *MM-psd*, *SQP*, *Cet* and *Seol* as the most significant of the six approaches). Looking at the values of RMSE, *SQP* clearly outperforms all the other approaches; however, it is inferior in all the other respects. Cardinality is, without exception, always set to the maximum value ( $m = 130$ ), and the  $L_1$  norm is visibly higher than for the other methods, aside from *Seol*. Values of the smoothness factor are far higher than for the other approaches, comparable only to that of *Seol*, hence it was necessary to zoom in on the region of the bottom left subplot in order to compare the other boxes. Finally, the computation time is much longer than that of *MM*, and since *SQP* is not straightforward to implement in parallel, this cost is irreducible. On the other side, *MM-psd* has a bit higher RMSE than *SQP*, yet still significantly lower than the previous state-of-the-art methods. At the same time, it keeps the cardinality relatively low, somewhere in between *Cet* and *Seol*, and the results are relatively smooth. We address the reader to check the supplementary video materials for a complete visual comparison of the animation sequences.

The only aspect where other methods exhibit better performance is execution time. However, since this method targets the production of movie and game animation, it is not restricted to real-time computations, and the performance speed in our results (around 10s per frame) is feasible. This is especially favorable when contrasted with the solution of a general-purpose solver (*SQP*), which is about 15 times slower. Note that here we implemented the algorithm sequentially, but due to the construction, *inner iterations* could be implemented in parallel to reduce the execution time further.

In Figure 1, we see an example frame for character *Omar*, with meshes and activation weights. Again we note that, while the mesh yield by *SQP* gives a flawless reconstruction of the reference mesh, it exhibits too high cardinality of the weights vector, and hence might affect the stability of the result and make posterior manual editing impossible. On the

	RMSE	Card.	L1	Smooth. factor	Time	$\alpha$
MM	0.0898	85.2	<b>9.10</b>	0.0034	11.29	5
SQP	<b>0.0325</b>	130.	12.4	0.0201	172.7	5
Cet	0.1671	95.4	9.38	<b>0.0029</b>	<b>0.002</b>	5
Seol	0.1625	<b>78.9</b>	12.7	0.0207	0.017	/

Table 1: Comparison of the obtained metrics values for the four selected approaches, for *Omar*, where *MM* is with *psd* initialization. The best value for each metric is given in bold. The last column gives an optimal value of a regularization parameter  $\alpha$ .

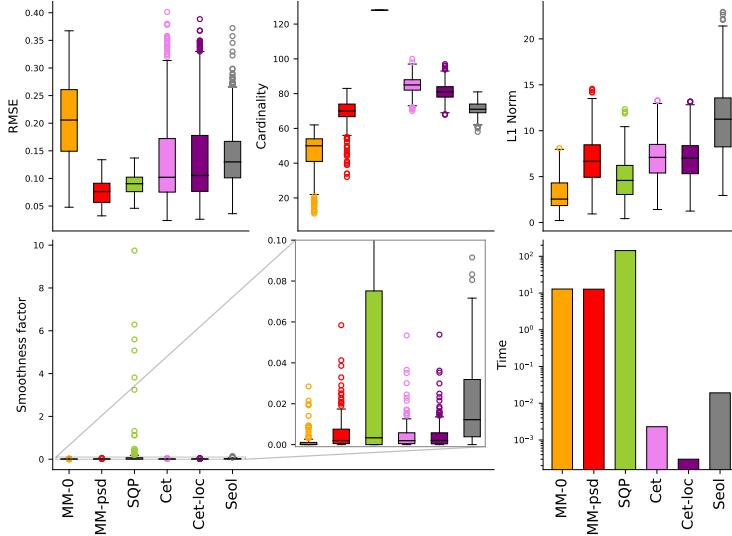


Figure 11: Values of the three metrics (mesh RMSE, weights cardinality, weights  $L_1$  norm, and temporal smoothness of weight curves) and execution time (in seconds) for *Danielle*. Execution time is presented in a log-scale, because of the wide range — for *Cet-loc* it takes 0.0003 s, and for *SQP* 143.2 s. See Table 2 for numerical details.

other side, the three methods under the linear blendshape approximation (*Cet*, *Cet-loc* and *Seol*) show visible misfits in the mesh — meshes of *Cet* and *Cet-loc* are very similar, and both are slightly off in the mouth corner region; notice that corners are not as widely spread as in the reference, that the upper lip covers a larger surface of the teeth and that the shadow under the bottom lip is not as visible. *Seol* exhibits low cardinality and lower average error than *Cet*, yet it gives a completely different mouth expression. Our method *MM* is the only one that gives both a good reconstruction of the mesh and low cardinality of the weight vector.

Metric values for *Danielle* are given in Figure 11 and Table 2. The conclusions are somewhat similar to the case of *Omar*. Here, *MM* actually gives lower RMSE than *SQP* (and any other approach), and even the cardinality is lower than that of *Seol*. Another notable difference is that *SQP* contrast even more with the other methods in term of smoothness, completely dominating the corresponding subfigure. The example frame with predictions for *MM-psd*, *Cet*, and *Seol* is presented in Figure 12. Both *Cet* and *Seol* show higher errors in the region around the mouth (as indicated by the red color), with *Cet* giving slightly less stretched mouth corners compared to the original, while *Seol* poorly reconstructs the lower lip — the bottom row of teeth is completely visible, while it is occluded in the reference frame. Our method

	RMSE	Card.	L1	Smooth. factor	Time	$\alpha$
MM	<b>0.0756</b>	<b>69.3</b>	6.72	0.0062	12.73	5
SQP	0.0894	128.	<b>4.64</b>	0.3390	143.2	5
Cet	0.1280	84.6	7.02	<b>0.0051</b>	<b>0.002</b>	5
Seol	0.1389	71.5	11.1	0.0212	0.019	/

Table 2: Comparison of the obtained metrics values for the four selected approaches, for *Danielle*. The best value for each metric is given in bold. The last column gives an optimal value of a regularization parameter  $\alpha$ .

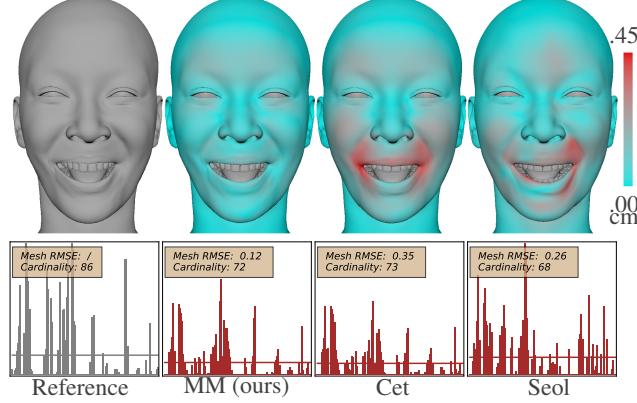


Figure 12: Example frame prediction for *Danielle*. The top row shows obtained meshes, while the bottom represents corresponding activations of the controller weights. Red tones in the meshes indicate a higher error of the fit, according to the color bar on the right. The average weight activation of each solution is indicated with a horizontal line. The average mesh error and cardinality (i.e. the number of non-zero weights) of the solution are given in a text box.

	RMSE	Card.	L1	Smooth. factor	Time	$\alpha$
MM	<b>0.0556</b>	69.0	4.48	0.0027	8.276	5
SQP	0.0809	114.	<b>2.66</b>	0.0118	164.3	5
Cet	0.0682	80.9	4.90	<b>0.0024</b>	<b>0.004</b>	5
Seol	0.1140	<b>65.4</b>	7.47	0.0181	0.015	/

Table 3: Comparison of the obtained metrics values for the four selected approaches, for *Myles*. The best value for each metric is given in bold. The last column gives an optimal value of a regularization parameter  $\alpha$ .

gives a mesh reconstruction that closely resembles the original, while the cardinality is comparable to the benchmark methods.

For *Myles*, boxplots are presented in Figure 13 (and mean values in Table 3). Our method *MM* gives the lowest RMSE and the cardinality that is in between that of *Cet* and *Seol*. In this case, *Cet* (and *Cet-loc*) gives quite low RMSE — in the median value, it is even lower than *SQP*, yet it varies much more, giving significantly larger upper quantiles. The example frame for *Myles* is in Figure 14. While the average RMSE is larger (and more red tones are visible in the mesh) for *Cet* than for *MM*, neither of the two produces visible flaws, and bar plots of the weight activations are very similar. On the other side, *Seol* shows a poor fit in the shapes of the lips and cheeks.

*Char 4* and *Char 5* are proprietary models, hence we are not showing the meshes; however, the results are summarized in Figures 15 and 16 and in Tables 4 and 5, respectively. As mentioned earlier, a face model of *Char 5* has a different structure, with many non-blendshape components, while our algorithm targets detailed and accurate blendshape rigs. However, these results show that even with relaxed assumptions about the face model, our method is comparable to state-of-the-art solutions. For *Char 4* the conclusions are similar to those for *Myles*, while for *Char 5* the main takeaway is that *MM* is slightly better than other methods in terms of RMSE, and significantly outperforms the benchmarks in terms of cardinality. A difference in the smoothness factor between the methods is not so drastic here, with only *Seol* showing considerably higher values.

	RMSE	Card.	L1	Smooth. factor	Time	$\alpha$
MM	<b>0.1031</b>	<b>85.0</b>	10.5	0.0098	10.01	5
SQP	0.1142	147.	<b>7.92</b>	0.0444	84.5	5
Cet	0.1606	99.5	10.8	<b>0.0088</b>	<b>0.009</b>	5
Seol	0.1440	87.1	15.8	0.0263	0.021	/

Table 4: Comparison of the obtained metrics values for the four selected approaches, for *Char 4*. The best value for each metric is given in bold. The last column gives an optimal value of a regularization parameter  $\alpha$ .

	RMSE	Card.	L1	Smooth. factor	Time	$\alpha$
MM	<b>0.0564</b>	<b>28.3</b>	2.74	0.1638	2.985	5
SQP	0.0745	62.0	<b>1.82</b>	0.2189	32.24	5
Cet	0.0611	39.8	3.26	<b>0.1550</b>	<b>0.002</b>	5
Seol	0.0834	<b>38.2</b>	5.68	0.4908	0.008	/

Table 5: Comparison of the obtained metrics values for the four selected approaches, for *Char 5*. The best value for each metric is given in bold. The last column gives an optimal value of a regularization parameter  $\alpha$ .

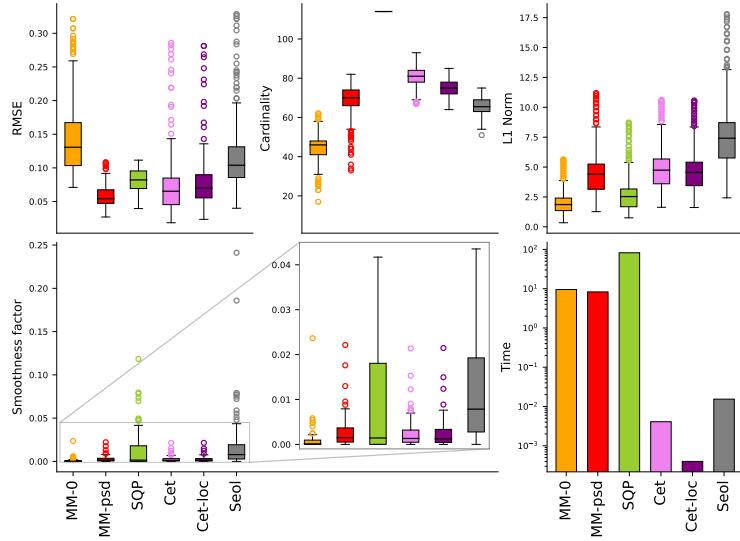


Figure 13: Values of the four metrics (mesh RMSE, weights cardinality, weights  $L1$  norm, and temporal smoothness of weight curves) and execution time (in seconds) for *Myles*. Execution time is presented in a log-scale, because of the wide range — for *Cet-loc* it takes 0.0004 s, and for *SQP* 164.3 s. See Table 3 for numerical details.

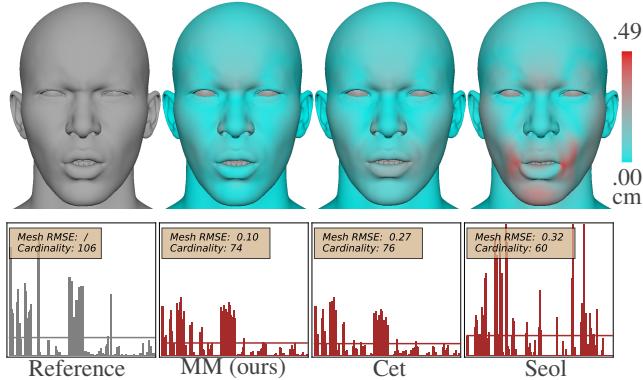


Figure 14: Example frame prediction for *Myles*. The top row shows obtained meshes, while the bottom represents corresponding activations of the controller weights. Red tones in the meshes indicate a higher error of the fit, according to the color bar on the right. The average weight activation of each solution is indicated with a horizontal line. The average mesh error and cardinality of the solution are given in a text box.

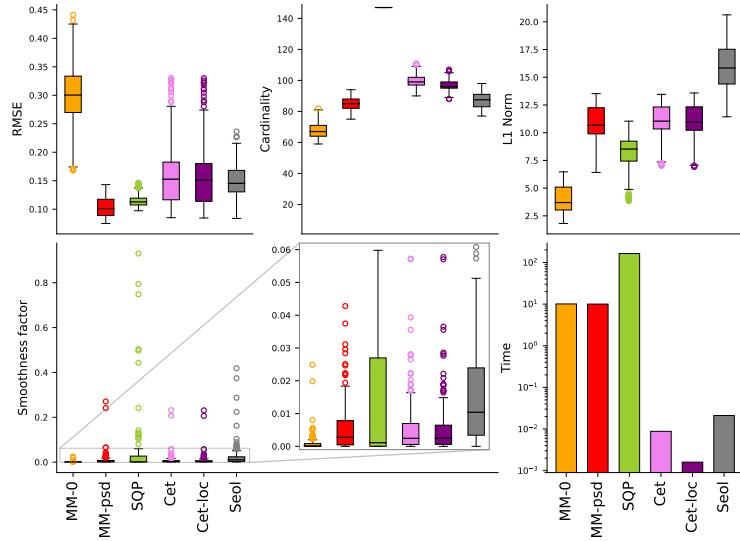


Figure 15: Values of the four metrics (mesh RMSE, weights cardinality, weights  $L1$  norm, and temporal smoothness of the weight curves) and execution time (in seconds) for *Char 4*. Execution time is presented in a log-scale, because of the wide range — for *Cet-loc* it takes 0.0016 s, and for *SQP* 84.5 s. See Table 4 for numerical details.

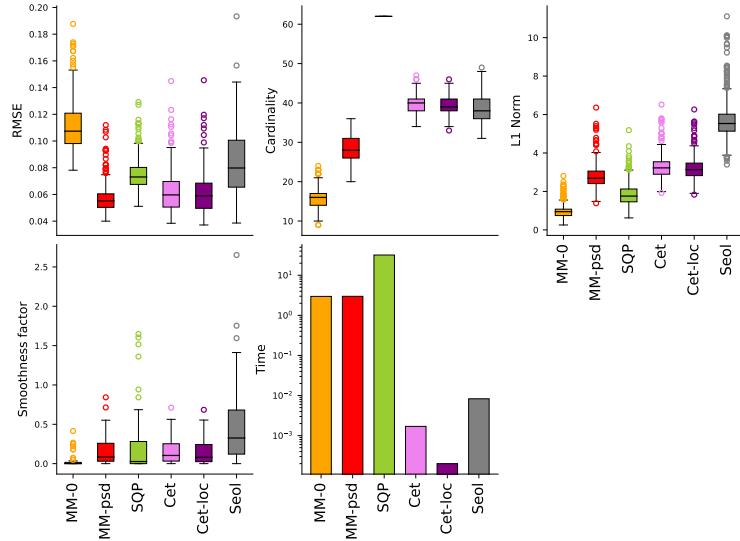


Figure 16: Values of the four metrics (mesh RMSE, weights cardinality, weights  $L1$  norm, and temporal smoothness of the weight curves) and execution time (in seconds) for *Char 5*. Execution time is presented in a log-scale, because of the wide range — for *Cet-loc* it takes 0.0002 s, and for *SQP* 32.24 s. See Table 5 for numerical details.

It is interesting to point out that, even though no temporal prior was included in the objective, *MM* and *Cet* yield smooth results for all five datasets, as indicated by the respective smoothness factor and also confirmed by the supplementary video materials. This was not the case for *SQP* or *Seol*. In Seol et al. [2011], the solution of *Seol* was modified afterward, using the graph simplification technique, to produce smooth animation curves. However, this was outside of the scope of our paper, hence we did not apply those additional steps.

#### 5.4 Animator Feedback

To make the analysis of the results complete, we include here also the feedback from the expert animators. Five animators were asked to rank the methods (*MM*, *SQP*, *Cet*, *Cet-loc*, *Seol*) from the best performing to the worst. The criterion was the appearance of the animated results compared to the reference motion — the animators were looking at the flip-tests provided as the supplementary video materials, except that the method names were masked to avoid possible bias.

The answers of the animators were heterogenous: sometimes the same animator would give different rankings of the methods for different animated characters (*Omar*, *Danielle*, *Myles*), sometimes there would be ties between two or more methods, and in some cases, the animator would only be able to tell the best and the worst of five results. This leaves us with a set of pairwise comparisons of the results. We will estimate the overall ranking of the five methods using the *Bradley-Terry model* Bradley and Terry [1952], Chen et al. [2013], as it is a common ranking procedure when the outcomes of the pairwise comparisons are available.

	MM	SQP	Cet	Cet-loc	Seol
MM	-	5	12	12	11
SQP	6	-	9	9	12
Cet	0	3	-	3	3
Cet-loc	0	2	2	-	3
Seol	2	3	5	5	-

Table 6: Matrix  $\mathbf{S}$  of the outcomes of pairwise comparisons between the five methods. A value in row  $i$  and column  $j$  tells how many times the  $i^{th}$  method ranked better than the  $j^{th}$ .

The Bradley-Terry model is an iterative procedure for estimating the relative strength of the objects (methods in our case) from the set of pairwise comparisons. The outcomes of the comparisons are collected into a square matrix  $\mathbf{S} \in \mathbb{R}^{N \times N}$ , as seen in Table 6; the value in row  $i$  and column  $j$  shows how many times the  $i^{th}$  object (method) ranked better than the  $j^{th}$ , while the value in a row  $j$  and column  $i$  indicates the number of the opposite outcomes. From Table 6, we can already see that the values in the first row are the highest, telling us that *MM* was most often ranked as the better method compared to the others, while *SQP* (the second row) is slightly behind. Also, we see that the values of the first column are relatively lower than for the others, confirming that *MM* rarely lost the pairwise comparisons.

Further, the vector of relative strengths of the methods,  $\mathbf{s} \in \mathbb{R}^N$ , is estimated iteratively, as follows. For each object  $i$ , the  $i^{th}$  coordinate of the vector is

$$s_i \leftarrow \frac{\sum_{j \neq i} S_{ij}}{\sum_{j \neq i} \frac{S_{ij} + S_{ji}}{s_i + s_j}}. \quad (23)$$

The final values of the relative strength vector  $\mathbf{s}$ , scaled so that it sums to 1, are presented in Figure 17. *MM* shows a clear advantage over the other methods, and it is followed by *SQP*. Other methods are far behind these two. While *SQP* was often showing a lower RMSE than *MM*, the ranking was more in favor of *MM*, and we can interpret this by the fact that *SQP* produces high cardinality, which led to less stable (or less smooth) transitions between the frames of animation; hence, the estimated animation sequences did not look as organic as *MM*, leading to the lower relative score assigned by the animators.

Some remarks that animators made, aside from the rankings, are that the differences are easiest to tell in the region of the mouth, especially when the corners are wide open, and that *MM* and *SQP* show a closer fit in this region. This is something that we could have expected since the two methods include the corrective terms when fitting, and the majority of the quadratic corrective terms are usually targeting the lips of the characters.

## 6 Conclusion

The method proposed in this paper applies the majorization minimization paradigm in order to easily achieve the solution to the problem of rig inversion even when the quadratic blendshape terms are utilized. Our method gives a better

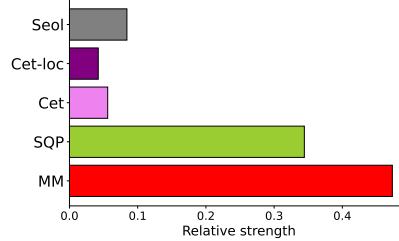


Figure 17: Bradley-Terry estimate of the relative strength vector  $\mathbf{s}$ . Methods that use corrective terms (*MM*, *SQP*) show a clear superiority over simpler methods.

fit in the details of the face mesh while not increasing the cardinality of the weight vector compared to state-of-the-art methods proposed in Seol et al. [2011] and Cetinaslan and Orvalho [2020a]; hence, the method is highly applicable in realistic face animation and targets the applications where accuracy is preferable to real-time execution such as the close shot animations in video games or movies production. It is further worth mentioning that the construction of the algorithm gives space for the parallel implementation of the inner iterations, and in future work, we will address this to reduce the execution time additionally. Another aspect that we will address in future research is to include an additional step of face segmentation. This might lead to distributed model and possibly even higher precision in fitting the fine details of the face mesh.

### Acknowledgements

The authors would like to thank 3Lateral studio for guidelines in the animation applications domain, and in specific a group of animators, Aleksa Bračić, Nikola Stošić, Đorđe Ilić, Igor Erić, and Lazar Damjanov, for the valuable feedback on the animated results. We want to thank Dr. Filipa Valdeira for sharing her expertise in the ranking methods to help us evaluate the animators’ feedback.

### Funding

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).

### References

- Ozan Cetinaslan and Veronica Orvalho. Sketching manipulators for localized blendshape editing. *Graphical Models*, 108:101059, 2020a. doi:<https://doi.org/10.1016/j.gmod.2020.101059>.
- Yeongho Seol, Jaewoo Seo, Paul Hyunjin Kim, J. P. Lewis, and Junyong Noh. Artist friendly facial animation retargeting. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. Association for Computing Machinery, 2011. doi:<https://doi.org/10.1145/2024156.2024196>.
- Byoungwon Choe, Hanook Lee, and Hyeong-Seok Ko. Performance-driven muscle-based facial animation. *The Journal of Visualization and Computer Animation*, 12(2):67–79, 2001. doi:<https://doi.org/10.1002/vis.246>.
- Eftychios Sifakis, Igor Neverov, and Ronald Fedkiw. Automatic determination of facial muscle activations from sparse motion capture marker data. In *ACM SIGGRAPH 2005 Papers*, page 417–425. Association for Computing Machinery, 2005. doi:<https://doi.org/10.1145/1186822.1073208>.
- Chenglei Wu, Derek Bradley, Markus Gross, and Thabo Beeler. An anatomically-constrained local deformation model for monocular face capture. *ACM Trans. Graph.*, 35(4):1–12, 2016. doi:<https://doi.org/10.1145/2897824.2925882>.
- Gaspard Zoss, Derek Bradley, Pascal Bérard, and Thabo Beeler. An empirical rig for jaw animation. *ACM Trans. Graph.*, 37(4):1–12, 2018. doi:<https://doi.org/10.1145/3197517.3201382>.
- Gaspard Zoss, Thabo Beeler, Markus Gross, and Derek Bradley. Accurate markerless jaw tracking for facial performance capture. *ACM Trans. Graph.*, 38(4):1–8, 2019. doi:<https://doi.org/10.1145/3306346.3323044>.
- Fratarcangeli Marco, Bradley Derek, A. Gruber, Zoss Gaspard, and Beeler Thabo. Fast nonlinear least squares optimization of large-scale semi-sparse problems. *Computer Graphics Forum*, 39, 2020. doi:<https://doi.org/10.1111/cgf.13927>.

- Fabian Hahn, Bernhard Thomaszewski, Stelian Coros, Robert W Sumner, and Markus Gross. Efficient simulation of secondary motion in rig-space. In *Proceedings of the 12th ACM SIGGRAPH/eurographics symposium on computer animation*, pages 165–171, 2013. doi:<https://doi.org/10.1145/2485895.2485918>.
- Alexandru Eugen Ichim, Ladislav Kavan, Merlin Nimier-David, and Mark Pauly. Building and animating user-specific volumetric face rigs. In *Symposium on Computer Animation*, pages 107–117, 2016.
- Alexandru-Eugen Ichim, Petr Kadlec, Ladislav Kavan, and Mark Pauly. Phace: Physics-based face modeling and animation. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.
- Yongli Hu, Baocai Yin, Yanfeng Sun, and Shiquan Cheng. 3d face animation based on morphable model. *Journal of Information and Computational Science*, 2(1):35–39, 2005.
- Michaël De Smet and Luc Van Gool. Optimal regions for linear model-based 3d face reconstruction. In *Asian conference on computer vision*, pages 276–289. Springer, 2010.
- Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. Real-time expression transfer for facial reenactment. *ACM Trans. Graph.*, 34(6), 2015. doi:<https://doi.org/10.1145/2816795.2818056>.
- Wang Mengjiao, Bradley Derek, Zafeiriou Stefanos, and Beeler Thabo. Facial expression synthesis using a global-local multilinear framework. In *Computer Graphics Forum*, volume 39, pages 235–245. Wiley Online Library, 2020. doi:<https://doi.org/10.1111/cgf.13926>.
- Frédéric H. Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and D. Salesin. Synthesizing realistic facial expressions from photographs. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, page 75–84, 1998. doi:<https://doi.org/10.1145/280814.280825>.
- Byoungwon Choe and Hyeong-Seok Ko. Analysis and synthesis of facial expressions with hand-generated muscle actuation basis. In *ACM SIGGRAPH 2006 Courses*. 2006. doi:<https://doi.org/10.1145/1198555.1198595>.
- Pushkar Joshi, Wen C Tien, Mathieu Desbrun, and Frédéric Pighin. Learning controls for blend shape based realistic facial animation. In *ACM SIGGRAPH 2006 Courses*. 2006. doi:[https://doi.org/10.1007/978-1-84628-907-1\\_9](https://doi.org/10.1007/978-1-84628-907-1_9).
- Zhigang Deng, Pei-Ying Chiang, Pamela Fox, and Ulrich Neumann. Animating blendshape faces by cross-mapping motion capture data. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, page 43–48. Association for Computing Machinery, 2006. doi:<https://doi.org/10.1145/1111411.1111419>.
- J.P. Lewis and Ken-ichi Anjyo. Direct manipulation blendshapes. *IEEE Computer Graphics and Applications*, 30(04):42–50, 2010. doi:<https://doi.org/10.1109/MCG.2010.41>.
- Hao Li, Thibaut Weise, and Mark Pauly. Example-based facial rigging. *ACM Trans. Graph.*, 29(4):1–6, 2010. doi:<https://doi.org/10.1145/1778765.1778769>.
- John P. Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Frederic H. Pighin, and Zhigang Deng. Practice and theory of blendshape facial models. *Eurographics (State of the Art Reports)*, 1(8):2, 2014. doi:<https://doi.org/10.2312/egst.20141042>.
- Ozan Cetinaslan and Veronica Orvalho. Stabilized blendshape editing using localized jacobian transpose descent. *Graphical Models*, 112:101091, 2020b. doi:<https://doi.org/10.1016/j.gmod.2020.101091>.
- Ko-Yun Liu, Wan-Chun Ma, Jackson Lee, Chuan-Chang Wang, and Chun-Fa Chang. Localized optimization for mocap-driven blendshapes. In *SCA 2010*, 2010.
- Jaewoo Seo, Geoffrey Irving, J. P. Lewis, and Junyong Noh. Compression and direct manipulation of complex blendshape models. *ACM Trans. Graph.*, 30(6):1–10, dec 2011. doi:<https://doi.org/10.1145/2070781.2024198>.
- Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi:[10.1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2).
- Richard H Byrd, Mary E Hribar, and Jorge Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.
- Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust region methods*. SIAM, 2000.
- Ying Sun, Prabhu Babu, and Daniel P Palomar. Majorization-Minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, 65(3):794–816, 2016. doi:<https://doi.org/10.1109/TSP.2016.2601299>.

- Yosra Marnissi, Emilie Chouzenoux, Amel Benazza-Benyahia, and Jean-Christophe Pesquet. Majorize-Minimize adapted Metropolis-Hastings algorithm. *IEEE Transactions on Signal Processing*, 68:2356–2369, 2020. doi:<https://doi.org/10.1109/TSP.2020.2983150>.
- Supplemental File not for review. A majorization minimization-based method for nonconvex inverse rig problems in facial animation: Algorithm derivation. 2022.
- Thomas Neumann, Kiran Varanasi, Stephan Wenger, Markus Wacker, Marcus Magnor, and Christian Theobalt. Sparse localized deformation components. *ACM Trans. Graph.*, 32(6):1–10, 2013. doi:<https://doi.org/10.1145/2508363.2508417>.
- Bindita Chaudhuri, Noranart Vesdapunt, Linda Shapiro, and Baoyuan Wang. Personalized face modeling for improved face reconstruction and motion retargeting. In *European Conference on Computer Vision*, pages 142–160. Springer, 2020. doi:[https://doi.org/10.1007/978-3-030-58558-7\\_9](https://doi.org/10.1007/978-3-030-58558-7_9).
- Hao Li, Jihun Yu, Yuting Ye, and Chris Bregler. Realtime facial animation with on-the-fly correctives. *ACM Trans. Graph.*, 32(4):42–1, 2013. doi:<https://doi.org/10.1145/2461912.2462019>.
- Roger Blanco i Ribera, Eduard Zell, J. P. Lewis, Junyong Noh, and Mario Botsch. Facial retargeting with automatic range of motion alignment. *ACM Trans. Graph.*, 36(4), 2017. doi:<https://doi.org/10.1145/3072959.3073674>.
- Juyong Zhang, Keyu Chen, and Jianmin Zheng. Facial expression retargeting from human to avatar made easy. *IEEE Transactions on Visualization and Computer Graphics*, 28(2):1274–1287, 2020. doi:<https://doi.org/10.1109/TVCG.2020.3013876>.
- Ju Hee Han, Jee-In Kim, Hyungseok Kim, and Jang Won Suh. Generate individually optimized blendshapes. In *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 114–120, 2021. doi:<https://doi.org/10.1109/BIGCOMP51126.2021.00030>.
- Yeongho Seol, Wan-Chun Ma, and John P Lewis. Creating an actor-specific facial rig from performance capture. In *Proceedings of the 2016 Symposium on Digital Production*, pages 13–17, 2016.
- Yeongho Seol and J. P. Lewis. Tuning facial animation in a mocap pipeline. In *ACM SIGGRAPH 2014 Talks*, SIGGRAPH ’14. Association for Computing Machinery, 2014. doi:<https://doi.org/10.1145/2614106.2614108>.
- Hirose Kei and Higuchi Tomoyuki. Creating facial animation of characters via mocap data. *Journal of Applied Statistics*, 39(12):2583–2597, 2012. doi:<https://doi.org/10.1080/02664763.2012.724391>.
- Yeongho Seol, J.P. Lewis, Jaewoo Seo, Byungkuk Choi, Ken Anjyo, and Junyong Noh. Spacetime expression cloning for blendshapes. *ACM Trans. Graph.*, 31(2), 2012. doi:<https://doi.org/10.1145/2159516.2159519>.
- Christos Ouzounis, Alex Kiliias, and Christos Mousas. Kernel projection of latent structures regression for facial animation retargeting. In *Proceedings of the 13th Workshop on Virtual Reality Interactions and Physical Simulations*, page 59–65, 2017. doi:<https://doi.org/10.2312/vrphys.20171084>.
- José Serra, Ozan Cetinaslan, Shridhar Ravikumar, Verónica Orvalho, and Darren Cosker. Easy generation of facial animation using motion graphs. In *Computer Graphics Forum*, volume 37, pages 97–111. Wiley Online Library, 2018.
- Clément Reverdy, Sylvie Gibet, and Caroline Larboulette. Optimal marker set for motion capture of dynamical facial expressions. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, page 31–36. Association for Computing Machinery, 2015. doi:<https://doi.org/10.1145/2822013.2822042>.
- Yasutaka Furukawa and Jean Ponce. Dense 3d motion capture for human faces. In *2009 IEEE conference on computer vision and pattern recognition*, pages 1674–1681. IEEE, 2009.
- Derek Bradley, Wolfgang Heidrich, Tiberiu Popa, and Alla Sheffer. High resolution passive facial performance capture. *ACM Trans. Graph.*, 29(4), 2010. doi:<https://doi.org/10.1145/1778765.1778778>.
- Thabo Beeler, Fabian Hahn, Derek Bradley, Bernd Bickel, Paul Beardsley, Craig Gotsman, Robert W. Sumner, and Markus Gross. High-quality passive facial performance capture using anchor frames. *ACM Trans. Graph.*, 30(4), 2011. doi:<https://doi.org/10.1145/2010324.1964970>.
- Daniel Holden, Jun Saito, and Taku Komura. Learning inverse rig mappings by nonlinear regression. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1167–1178, 2016. doi:<https://doi.org/10.1109/TVCG.2016.2628036>.
- Stephen W. Bailey, Dalton Omens, Paul Dilorenzo, and James F. O’Brien. Fast and deep facial deformations. *ACM Trans. Graph.*, 39(4), 2020. doi:<https://doi.org/10.1145/3386569.3392397>.
- Steven L. Song, Weiqi Shi, and Michael Reed. Accurate face rig approximation with deep differential subspace reconstruction. *ACM Trans. Graph.*, 39, 2020. doi:<https://doi.org/10.1145/3386569.3392491>.

- Kim Seonghyeon, Jung Sunjin, Seo Kwanggyoon, Blanco I. Ribera Roger, and Noh Junyong. Deep learning-based unsupervised human facial retargeting. *Computer Graphics Forum*, 40, 2021. doi:<https://doi.org/10.1111/cgf.14400>.
- Jaewon Song, Byungkuk Choi, Yeongho Seol, and Jun yong Noh. Characteristic facial retargeting. *Computer Animation and Virtual Worlds*, 22, 2011. doi:<https://doi.org/10.1002/cav.414>.
- Daniel Holden, Jun Saito, and Taku Komura. Learning an inverse rig mapping for character animation. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 165–173, 2015. doi:<https://doi.org/10.1145/2786784.2786788>.
- Wei-Wen Feng, Byung-Uck Kim, and Yizhou Yu. Real-time data driven deformation using kernel canonical correlation analysis. *ACM Trans. Graph.*, 27(3):1–9, 2008. doi:<https://doi.org/10.1145/1360612.1360690>.
- Hui Yu and Honghai Liu. Regression-based facial expression optimization. *IEEE Transactions on Human-Machine Systems*, 44(3):386–394, 2014. doi:<https://doi.org/10.1109/THMS.2014.2313912>.
- Sofien Bouaziz, Yangang Wang, and Mark Pauly. Online modeling for realtime facial animation. *ACM Trans. Graph.*, 32(4), 2013. doi:<https://doi.org/10.1145/2461912.2461976>.
- Cumhur Ozan Çetinaslan. *Position Manipulation Techniques for Facial Animation*. PhD thesis, Faculdade de Ciencias da Universidade do Porto, 2016.
- Jaewon Song, Roger Blanco i Ribera, Kyungmin Cho, Mi You, John P Lewis, Byungkuk Choi, and Junyong Noh. Sparse rig parameter optimization for character animation. In *Computer Graphics Forum*, volume 36, pages 85–94. Wiley Online Library, 2017.
- Joonho Kim and Karan Singh. Optimizing ui layouts for deformable face-rig manipulation. *ACM Trans. Graph.*, 40(4):1–12, 2021.
- Li Zhang, Noah Snavely, Brian Curless, and Steven M Seitz. Spacetime faces: high resolution capture for modeling and animation. In *ACM SIGGRAPH 2004 Papers*, pages 548–558. 2004.
- Ken Anjyo, Hideki Todo, and John P Lewis. A practical approach to direct manipulation blendshapes. *Journal of Graphics Tools*, 16(3):160–176, 2012.
- Kyung-Gun Na and Moon-Ryul Jung. Local shape blending using coherent weighted regions. *The Visual Computer*, 27(6–8):575, 2011. doi:<https://doi.org/10.1007/s00371-011-0591-x>.
- J. Rafael Tena, Fernando De la Torre, and Iain Matthews. Interactive region-based linear 3D face models. In *ACM SIGGRAPH 2011 Papers*. Association for Computing Machinery, 2011. doi:<https://doi.org/10.1145/1964921.1964971>.
- Doug L. James and Christopher D. Twigg. Skinning mesh animations. *ACM Trans. Graph.*, 24(3):399–407, 2005. doi:<https://doi.org/10.1145/1073204.1073206>.
- Marco Romeo and S. Schwartzman. Data-driven facial simulation. In *Computer Graphics Forum*, volume 39, pages 513–526. Wiley Online Library, 2020. doi:<https://doi.org/10.1111/cgf.14089>.
- Stevo Racković, Cláudia Soares, Dušan Jakovetić, Zoranka Desnica, and Relja Ljubobratović. Clustering of the blendshape facial model. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 1556–1560, 2021. doi:<https://doi.org/10.23919/EUSIPCO54536.2021.9616061>.
- Mark Schmidt. Least squares optimization with 11-norm regularization. *CS542B Project Report*, 504:195–221, 2005.
- Stephen Wright, Jorge Nocedal, et al. Numerical optimization. *Springer Science*, 35(67–68):7, 1999.
- Elizabeth D. Schifano, Robert L. Strawderman, and Martin T. Wells. Majorization-Minimization algorithms for nonsmoothly penalized objective functions. *Electronic Journal of Statistics*, 4:1258–1299, 2010. doi:<https://doi.org/10.1214/10-EJS582>.
- Ranganathan Ananth. The levenberg-marquardt algorithm. 2004.
- Zhixin Fang, Libai Cai, and Gang Wang. Metahuman creator the starting point of the metaverse. In *2021 International Symposium on Computer Technology and Information Science (ISCTIS)*, pages 154–157. IEEE, 2021. doi:<https://doi.org/10.1109/ISCTIS51085.2021.00040>.
- J. Rafid Siddiqui. Fexgan-meta: Facial expression generation with meta humans. 2022.
- Keenan Crane, Clarisse Weischedel, and Max Wardetzky. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)*, 32(5):1–11, 2013.
- Autodesk, INC. Maya, 2019-01-15. URL <https://autodesk.com/maya>. 2019.
- Peter J Green and Bernard W Silverman. *Nonparametric regression and generalized linear models: a roughness penalty approach*. Crc Press, 1993.

Even Mehlm and Christian Tarrou. Invariant smoothness measures for surfaces. *Advances in Computational Mathematics*, 8(1):49–63, 1998.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 193–202, 2013.