Accurate Solution of the Inverse Rig for Realistic and Complex Blendshape Models

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

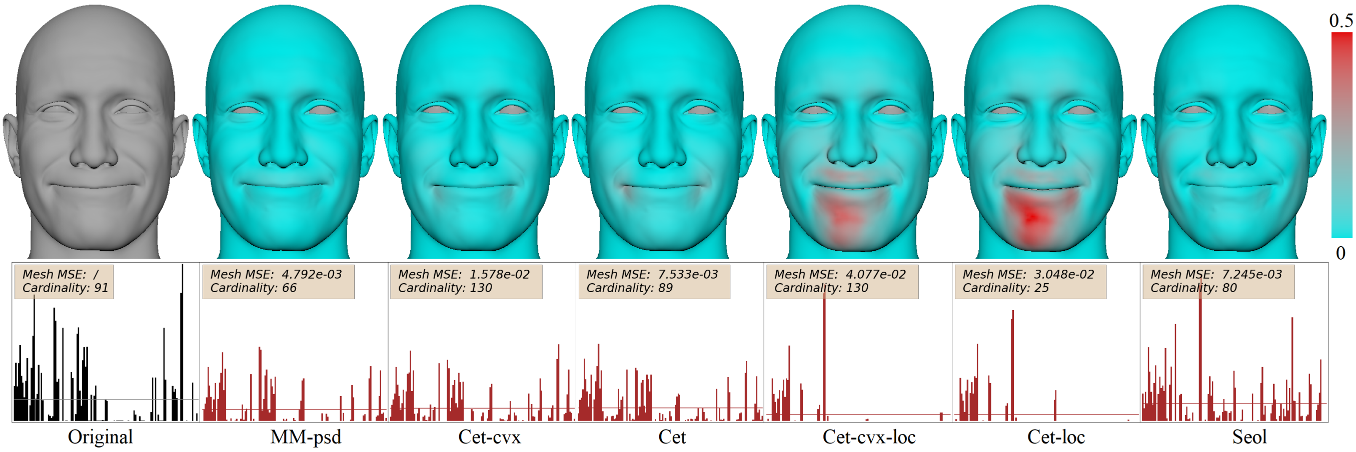

Fig. 1. The original frame mesh and the estimations of our method (*MM-psd*) versus the linear approaches proposed by [1] (*Cet-cvx* for the case when constraints are added in the problem formulation and **Cet** when the resulting vectors are clipped afterwards to satisfy the feasibility conditions; as well as localized approximations, *Cet-loc-cvx* and *Cet-loc*, under the analogous conditions) and [2] (Seol). The top row shows obtained meshes, while the bottom represents corresponding activations of the controller weights. Red tones in the meshes indicate higher error of the fit, accordint the colorbar on the right. The average weight activation of each solution is indicated with a horizontal line. Average mesh error and cardinality of the weight vector are given for each method, and we aim for the lowest error while keeping the cardinality relatively low. Note that two localized approaches *Cet-loc* and *Cet-loc-cvx* give inaccurate expressions. Our method *MM-psd* is further outperforming the others as it better resembles the original in the region under the lower lip and the wrinkles around the cheeks and at the same time has lower number of activated weights.

***Abstract* – We propose a new model-based algorithm for solving the inverse rig problem in the facial animation, exhibiting higher accuracy of the fit and lower cardinality of the weights compared to state-of-the-art methods. 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 and find the optimal solution using majorization minimization procedure. Unlike the prior solutions, our algorithm goes beyond a linear blendshape model, and employs the quadratic corrective terms, necessary for correctly fitting fine details of the mesh. We show results using both proprietary and open-source animated characters of a high quality and level of details. Our algorithm is benchmarked with several stat-of-the-art approaches, and shows an overall superiority or the results.**

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

I. Introduction

Facial animation is a popular research topic in academia as well as in the industry due to its complexity and the importance of facial expressions in our perception of other people. The most common model for animating the faces is a blendshape rig because it provides intuitive controls. Automated estimation of blendshape weights from the motion capture, known as the inverse rig problem, allows faster and cheaper animation. The solution of an inverse problem needs to have a high mesh fidelity because we are sensitive even to subtle changes of expression hence poor retargeting might easily produce the uncanny valley effect and repel the user. Additionally, the solution vector should be sparse in order to keep the animation editable (if needed), and also because a high number of activated components could produce artifacts in the mesh even if the constraints are satisfied (see Figure 2). We make distincion between *data-based* and *modle-based* approaches for solving the inverse rig — the former assume a rig function as a black-box and demand long animation sequences that span a whole space of expected expressions in order to train a good regressor, while the latter only demand

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

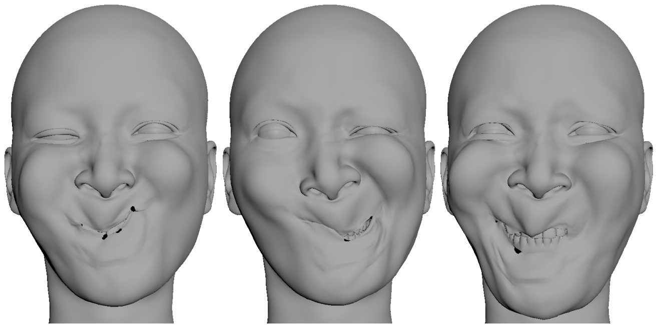

Fig. 2. Examples of dense activation vectors. All the activation weights of this model are set to a random value within a feasible interval $[0,1]$, and we can see that there are many anatomically incorrect deformations as well as breaking of the mesh.

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; 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 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, and it exhibits a relative improvement of 24% in mesh error while at the same time reducing cardinality of the weight vector by 25%.

A. Contributions

To the best of our knowledge, this paper is the first one to propose application of quadratic corrective terms of the blendshape model in solving the inverse rig problem. In particular, we devise a surrogate function that allows for easy solution of the inverse rig under the assumptions of quadratic terms and constraints over the weight vector. The obtained solution produces high-accuracy mesh fit and low cardinality of the estimated weight vector, while the execution time is feasible, hence it is suitable for realistic face animation in the movie and gaming industry.

This paper presents the solution for inverse rig and quadratic blendshapes from a domain point of view, including the explanation how the method works and a comprehensive set of experiments on proprietary and open source state-of-the-art animation characters. In a companion paper [3] we present a detailed method's derivation from the optimization theory perspective and convergence analysis.

B. Notation

Throughout this paper scalar values will be denoted with lowercase Latin a,b,c, or lowercase Greek α,β,γ letters. Vectors are denoted with bold lowercase $\mathbf{a},\mathbf{b},\mathbf{c}$ and are indexed using a subscript, i.e., the i^{th} element of a 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 $(\mathbf{b}_0,\mathbf{b}_1,...,\mathbf{b}_m)$ 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 area assumed to be column vectors, and $[a_1,...,a_n]$ 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 is not indexing, but a specific matrix corresponding to the (vertex) position i. A notation $\mathbf{A}=[\mathbf{a}_1,...,\mathbf{a}_n]$ means that a matrix $\mathbf{A}$ is obtained by stacking vectors $\mathbf{a}_1,...,\mathbf{a}_n$ along the columns. Functions are given using lowercase Latin or Greek characters, but always with a corresponding parameters in the brackets $a(\cdot),b(\cdot),\alpha(\cdot),\beta(\cdot)$. A set of real numbers and a set of positive integers are given by $\mathbb{R}$ and $\mathbb{N}$, respectively. Euclidean norm is denoted by $\|\cdot\|$.

II. Related Work

The blendshape animation has been a research topic for more than two decades [4]–[6]. The main components of a model are a neutral face mesh, a blendshape basis (local deformations of a neutral face), and a set of controllers corresponding to the meshes from the basis. A blendshape basis is often sculpted by hand, but since this task demands a lot of time and effort there are numerous papers proposing automated solutions for creating blendshapes. Two main approaches are considered — building a basis from a dense set of captured data [7]–[10] or deforming a generic set of blendshapes to build personalized blendshape meshes [11]–[15].

Animation can be obtained, once the blendshape basis is created, by adjusting the activation weights. This can be done manually or automatically if there is a reference motion. The latter is called automatic keyframe animation, or inverse rig problem, and it is the main focus of our paper. Reference motion is a (sparse or dense) set of markers recorded from an actor's face using motion capture (MoCap) systems [16], [17]. Sparse set of markers is a common approach, particularly if the motion should be retargeted to a fantasy character with face significantly different from the source actor [18]–[21], and it demands special care in positioning the markers on both source and target faces [22]. Although this technique is sufficient for general purpose MoCap, it fails to capture fine details of the face. For this reason markerless methods are developed to provide a high fidelity performance capture [23]–[25]. 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 [26]–[29], radial basis functions [16], [17], [30], [31] or other forms of regression [32]–[34]. 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 linear rigs to fit the acquired mesh, yielding convex optimization problems [5], [11], [18], [35]. We specifically target realistic facial animation with a high level of detail, and for this reason, we need to go beyond linear rigs and include quadratic corrective terms, as studied in [26], [36].

One generalization of inverse rig learning is the problem of correcting a solution that facilities direct manipulation, allowing an artist to refine the final rig by dragging specific vertices of the face directly and producing the desired expression [1], [2], [37]–[39]. In this case, solutions must be available in real-time hence methods are often restricted to a sparse set of markers or even a single vertex [37]. In this paper we are not concerned with real-time execution but aim for more precise mesh fit, hence we assume models with large number of vertices.

A possible approach towards 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 [5]. More recent papers model complex splits, either manually [2], [40], semi-automatically [41]–[43] or automatically [19], [22], [27], [36], [44], [45]. Clustering based on the underlying deformation model has been considered in [46] and [47], where a 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. While the algorithm proposed here offers a certain level of parallelization, we do not focus on distributed models, and all the experiments are performed sequentially over a nonsegmented face mesh.

Detailed introductions to principles of blendshape animation can be found in references [48] and [35].

III. Rig Approximation and Inverse Rig

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

A. Linear Blendshape Model

A *blendshape model* consists of a neutral face $\mathbf{b}_0 \in \mathbb{R}^{3n}$ and a set of m blendshape vectors $\mathbf{b}_1,...,\mathbf{b}_m \in \mathbb{R}^{3n}$ that represent atomic expressions obtained by local deformations over $\mathbf{b}_0$ (see Figure 3). Each blendshape $\mathbf{b}_i$ is assigned an activation parameter w_i that usually (but not exclusively) takes values between 0 and 1 [20]. A *linear delta blendshape function* $f_L(\cdot): \mathbb{R}^m \to \mathbb{R}^{3n}$ maps activation weights $w_1,...,w_m$ onto a mesh space, and it is defined as

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

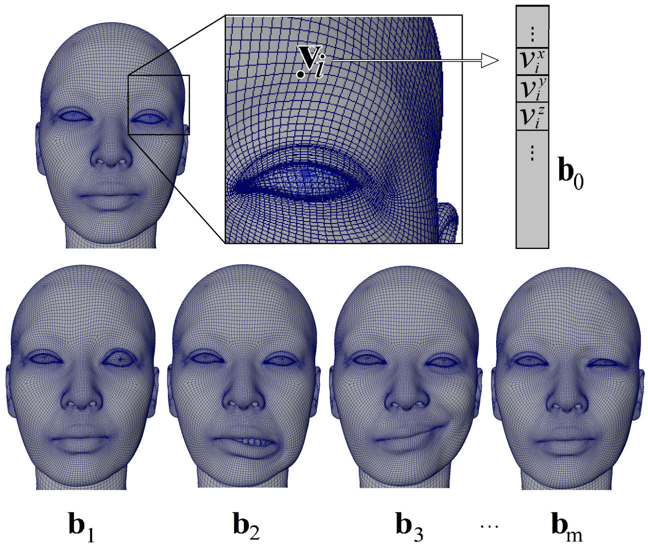

Fig. 3. Vectorization of meshes. Neutral mesh $\mathbf{b}_0$ on top, and example blendshapes below. Each face vertex $\mathbf{v}_i$ for $i = 1,...,n$ is unravel into a vector of coordinates x, y, z and those coordinate-vectors are stack into a single blendshape vector.

where $\Delta\mathbf{b}_i = \mathbf{b}_i - \mathbf{b}_0$ for $i = 1,...,m$. If we collect blendshape vectors into a matrix $\mathbf{B} = [\Delta\mathbf{b}_1,...,\Delta\mathbf{b}_m]$, $\mathbf{B} \in \mathbb{R}^{3n\times m}$, a function can be written in a matrix form as

$$f_L(\mathbf{w}) = \mathbf{b}_0 + \mathbf{B}\mathbf{w}, \tag{2}$$

where $\mathbf{w} = [w_1,...,w_m]$ represents the vector of blendshape weights.

B. Quadratic Blendshape Model

In modern animation, with increasing level of detail and with avatars that closely resemble an actor (or a user), linear models are too simple and fail to span a desired space of motion. For this reason, additional *corrective shapes* are included [38], [48], and these are usually more numerous than base vectors. In particular the quadratic corrective terms are very common, and adding them on top of a linear function (2) 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 an artifact when activated together, so the corrective term is defined as $\mathbf{b}^{\{i,j\}} = \widehat{\mathbf{b}}^{\{i,j\}} - (\mathbf{b}_0 + \mathbf{b}_i + \mathbf{b}_j)$, where $\widehat{\mathbf{b}}^{\{i,j\}}$ represents a desired result of joint activation of deformers i and j (and 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 contribuition 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* $f_Q(\cdot): \mathbb{R}^m \to \mathbb{R}^{3n}$ can now be defined as

$$f_Q(\mathbf{w}) = \mathbf{b}_0 + \mathbf{B}\mathbf{w} + \sum_{(i,j)\in\mathscr{P}} w_i w_j \mathbf{b}^{\{i,j\}}, \tag{3}$$

where $\mathscr{P}$ represents a set of tuples (i, j) such that there is a quadratic corrective term between corresponding blendshapes

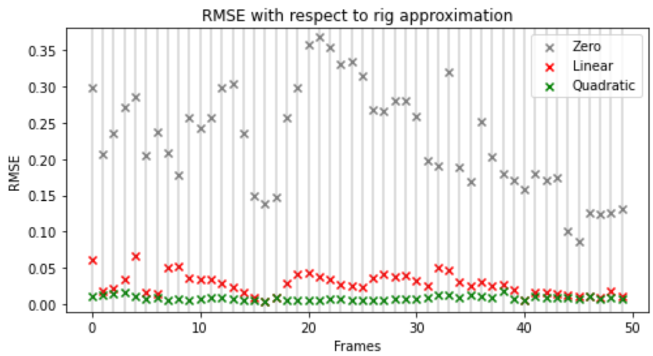

Fig. 4. RMSE between actual mesh and meshes obtained using different approximations.

$\mathbf{b}_i$ and $\mathbf{b}_j$. In Figure 4 we compare a root mean squared error (RMSE) between ground-truth animation frames and meshes obtained via linear and via quadratic approximation of a blendshape rig (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) for *Char 4*.

C. Inverse Rig

The *inverse rig*, or *automatic keyframe animation*, is the problem of finding optimal activation parameters to produce a target mesh $\widehat{\mathbf{b}}$, that 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:

$$\underset{\mathbf{w}}{\text{minimize}} \, \|f(\mathbf{w}) - \widehat{\mathbf{b}}\|^2, \tag{4}$$

where $f(\mathbf{w}) : \mathbb{R}^m \to \mathbb{R}^{3n}$ is a rig function, $\widehat{\mathbf{b}} \in \mathbb{R}^{3n}$ 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, and it is much harder if a large number of blendshapes is already activated [2].

The majority of papers pose the optimization problem similar to the form given in [1], utilizing linear blendshape models:

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

with $\alpha > 0$. Note that a neutral face $\mathbf{b}_0$ is omitted, hence the target $\widehat{\mathbf{b}}$ 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.

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

$$\begin{aligned} \text{step 1:} \quad & w_i = \underset{w}{\operatorname{argmin}} \, \|\mathbf{b}_i w - \widehat{\mathbf{b}}\|^2 \\ \text{step 2:} \quad & \widehat{\mathbf{b}} \leftarrow \widehat{\mathbf{b}} - \mathbf{b}_i w_i. \end{aligned} \tag{6}$$

The order in which controllers 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 controllers [2], [48].

All the methods available in the literature solve the inverse rig problem using a linear blendshape function because 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, and a linear model does not exhibit high-enough accuracy for this purpose. In the next section, we introduce a method that allows the application of a quadratic blendshape rig function (3) to produce an accurate and sparse solution of an inverse rig problem.

IV. Proposed Method

While a linear blendshape function is convenient for general inverse rig problems, and especially for cartoonish characters whose face proportions differ significantly from a source actor, 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 for higher accuracy compared to what a linear function offers, hence we consider a quadratic blendshape function (3). Additionally, we assume that controller weights $w_1, \dots, w_m$ must stay within $[0,1]$ interval. We propose a formulation of the optimization problem that takes into account all the above assumptions:

$$\underset{\mathbf{0} \leq \mathbf{w} \leq \mathbf{1}}{\text{minimize}} \, \|f_Q(\mathbf{w}) - \widehat{\mathbf{b}}\|^2 + \alpha \mathbf{1}^T \mathbf{w}. \tag{7}$$

The regularization term in (7), with $\alpha > 0$, is invoked to make a solution sparse. 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 [49]. We proceed in an *optimal increment* fashion, which means that 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

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

A quadratic term in the rig function makes this problem too complex to solve exactly, so we apply *majorization minimization* [50], [51] paradigm that solves an approximate problem. We introduce an upper bound function $\psi(\mathbf{v};\mathbf{w}) : \mathbb{R}^m \to \mathbb{R}$ over the original objective (8), such that $\|f_Q(\mathbf{w}+\mathbf{v}) - \widehat{\mathbf{b}}\|^2 + \alpha\mathbf{1}^T\mathbf{v} \leq \psi(\mathbf{v};\mathbf{w})$ 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

$$\underset{-\mathbf{w} \leq \mathbf{v} \leq \mathbf{1}-\mathbf{w}}{\text{minimize}} \, \psi(\mathbf{v};\mathbf{w}). \tag{9}$$

Many papers 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.

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 (9) instead of the original cost function in (8). The surrogate function (a global upper bound on the cost function in (8)) is carefully constructed such that it represents a good, $\mathbf{w}$-dependent approximation of the cost in (8) around the current point $\mathbf{w}$, and such that (9) is easy to solve.

In the rest of this paper we will write $\psi(\mathbf{v})$ instead of $\psi(\mathbf{v};\mathbf{w})$ for the sake of simplicity. The mathematical details of the derivation of functions $\psi(\cdot)$ are given in [3] and here we only give a final form:

$$\psi(\mathbf{v}) = \sum_{i=1}^{n}\Bigg(g_i^2 + 2g_i\sum_{j=1}^{m} h_{ij}v_j + 2\left(g_i\lambda_M(\mathbf{D}^{(i)},g_i) + \|\mathbf{h}_i\|^2\right)\sum_{j=1}^{m} v_j^2 + 2m\sigma^2(\mathbf{D}^{(i)})\sum_{j=1}^{m} v_j^4\Bigg) + \alpha\mathbf{1}^T\mathbf{v}, \tag{10}$$

where $\mathbf{D}^{(i)}$ are matrices obtained from corrective terms for each vertex i as $D_{jk}^{(i)} = D_{kj}^{(i)} = 1/2b_i^{\{j,k\}}$; terms $g_i = \mathbf{B}_i\mathbf{w} + \mathbf{w}^T\mathbf{D}^{(i)}\mathbf{w} - \widehat{b}_i$ and $\mathbf{h}_i = \mathbf{B}_i + 2\mathbf{w}^T\mathbf{D}^{(i)}$ are introduced to simplify the expression; $\sigma(\mathbf{D}^{(i)})$ is the largest singular value of a matrix $\mathbf{D}^{(i)}$; $\lambda_{\max}(\mathbf{D}^{(i)})$ and $\lambda_{\min}(\mathbf{D}^{(i)})$ are the largest and the smallest eigenvalues of matrix $\mathbf{D}^{(i)}$; and function $\lambda_M : (\mathbb{R}^{m\times m},\mathbb{R}) \to \mathbb{R}$ is defined as

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

The upper bound (10) allows solving the problem (9) in a closed form for each component separately. If we consider a single controller index $j \in \{1,...,m\}$ and regroup the coefficients of the bound function as $r = 2\sum_{i=1}^{3n}(g_i\lambda_M(\mathbf{D}^{(i)},g_i) + \|\mathbf{h}_i\|^2)$, $s = 2m\sum_{i=1}^{3n}\sigma^2(\mathbf{D}^{(i)})$, $q = 2\sum_{i=1}^{3n} g_ih_{ij} + \alpha$, (note that coefficient q is component-dependent, while r and s stay the same for all $j = 1,...,m$) we can write an objective function in a form of a quartic equation, without a cubic term:

$$\begin{aligned}&\underset{v_j}{\text{minimize}}\; qv_j + rv_j^2 + sv_j^4,\\ &\text{s.t. } 0 \leq w_j + v_j \leq 1.\end{aligned} \tag{11}$$

This procedure is summarized in Algorithm 1 and we refer to it as an *inner iteration*. Algorithm 1 solves the problem (11) for each component to give a full increment vector $\mathbf{v}$, that is, solving (10) is equivalent to solving (11) for each $j = 1,...,m$ independently. In this sense our approach is somewhat similar to the solution of [2], as given in (6); 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 allows for a parallel implementation of the procedure.

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

Algorithm 1 Inner Iteration

Require: Blendshape matrix $\mathbf{B} \in \mathbb{R}^{3n\times m}$, corrective blendshape matrices $\mathbf{D}^{(i)} \in \mathbb{R}^{m\times m}$ for $i = 1,...,3n$, target mesh $\widehat{\mathbf{b}} \in \mathbb{R}^{3n}$, regularization parameter $\alpha > 0$ and weight vector $\mathbf{w} \in [0,1]^m$.
Ensure: An optimal increment vector $\hat{\mathbf{v}}$ as a solution to (11).
  Compute coefficients q,r and s from eq. (11) 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 $j = 1,...,m$ **do**
    $q = 2\sum_{i=1}^{3n} g_ih_{ij} + \alpha$
    $\hat{v}_j = \text{argmin}_v\, qv + rv^2 + sv^4$
      s.t. $-w_j \leq v \leq 1 - w_j$
  end for
  return $\hat{\mathbf{v}}$

estimate $\mathbf{w}$ of the solution to (8), as explained in Algorithm 2, and after each iteration $t = 1,...,T$ we update the weight vector as $\mathbf{w}_{(t+1)} = \mathbf{w}_{(t)} + \mathbf{v}_{(t)}$. Initial vector can be chosen anywhere within the feasible space $\mathbf{0} \leq \mathbf{w}_{(0)} \leq \mathbf{1}$, but in Section V.we mention strategies for initialization based on domain knowledge, that lead to faster convergence and yield better results.

Algorithm 2 Proposed Method

Require: Blendshape matrix $\mathbf{B} \in \mathbb{R}^{3n\times m}$, corrective blendshapes $\mathbf{b}^{\{i,j\}} \in \mathbb{R}^{3n}$ for $(i,j) \in \mathscr{P}$, target mesh $\widehat{\mathbf{b}} \in \mathbb{R}^{3n}$, regularization parameter $\alpha > 0$, initial weight vector $\mathbf{w}_{(0)} \in [0,1]^m$, maximum number of iterations $T \in \mathbb{N}$ and the tolerance $\varepsilon > 0$.
Ensure: $\widehat{\mathbf{w}}$ - an approximate minimizer of the problem (7).
  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 $i = 1,...,3n$ **do**
    for $(j,k) \in \mathscr{P}$ **do**
      $D_{jk}^{(i)} = D_{kj}^{(i)} = 1/2b_i^{\{j,k\}}$.
    end for
    $\mathbf{D}^{(i)} \to \lambda_{\min}(\mathbf{D}^{(i)})$, $\lambda_{\max}(\mathbf{D}^{(i)})$, $\sigma(\mathbf{D}^{(i)})$.
  end for
  for $t = 0,...,T$ **do**
    Compute optimal increment $\hat{\mathbf{v}}$ using Algorithm 1
    Update the weight vector $\mathbf{w}_{(t)}$:
    $\mathbf{w}_{(t+1)} = \mathbf{w}_{(t)} + \hat{\mathbf{v}}$
    Check convergence
    if $|\psi(\hat{\mathbf{v}}) - \psi(\mathbf{0})| < \varepsilon$ **then**
      $\hat{\mathbf{w}} \leftarrow \mathbf{w}_{(t+1)}$
      return $\hat{\mathbf{w}}$
    end if
  end for
  $\hat{\mathbf{w}} \leftarrow \mathbf{w}_{(t+1)}$
  return $\widehat{\mathbf{w}}$

V. Evaluation

As mentioned earlier, we consider realistic human characters with high level of detail and controller weights

Fig. 5. Head models available at `MetaHumans`

restricted to lie between 0 and 1. The first three characters that we present in these results are publicly available within the `MetaHumans` platform — *Omar, Danielle* and *Myles*, as shown in Figure 5. The additional two datasets that we used to evaluate the method are property of 3Lateral studio — *Char 4* and *Char 5*. All the characters are accompanied with a short animation sequence covering a wide range of facial expressions, that was used to evaluate the methods. We exclude inactive vertices and the vertices in the neck and shoulder region 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, number of blendshapes differs (ranging between 60 and 150), and that means that different choice of regularization parameter $\alpha > 0$ (Eq. (5) and (7)) 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 controllers that are not based on a blendshape deformation (rotational and joint-like deformers), but we include it to show that our algorithm is robust enough to produce satisfying results even in this case.

As mentioned in Section Ç a state-of-the-art representative of a model-based approaches to solving the inverse rig problem is a method given by Cetinaslan [1], in Eq. (5). However, we have a problem applying this method directly because our animation models do not allow for setting weights outside of $[0,1]$ interval. The above approach does not consider any constraints over a weight vector. For this reason, we need to modify an obtained solution in order to evaluate it with our data. One possible approach is solving (5) directly, and afterward clipping the values of weight vector $\mathbf{w}$ that are negative or larger than 1. From now on, we refer to this approach as *Cet*. The other possibility is to include constraints $0 \leq w_j \leq 1$ for $j = 1,...,m$ in problem (5). We solve this using `CvxPy` [54]; hence we refer to this approach as *Cet-cvx* in the rest of this paper.

In the same paper [1], the authors propose a modification of the solution using a heat kernel to transform an original blendshape matrix $\mathbf{B}$ into a sparse approximation $\mathbf{B}^{loc}$. The idea is that vertices of the face should not affect controllers whose main impact is localized in a distant face region. Hence, this method provides localized and more stable results. The problem is posed identically to (5) except that matrix $\mathbf{B}$ is substituted with $\mathbf{B}^{loc}$. This means that again we have the same problem with constraints as above, so we consider approaches analogous to the two above and refer to them as *Cet-loc* and *Cet-loc-cvx* respectively.

A different approach is proposed by Seol [2] where the weights are optimized sequentially (6), starting from the ones that have a larger overall effect on the face. Again, we have to include constraints over the weight vector, so we include them in each algorithm iteration. The method (6) will be denoted as *Seol*. Note that in this approach there is no regularization parameter.

The results of our algorithm will be denoted as *MM* (*majorization-minimization*). In Section IV 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 rig 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 approach as *MM-psd*. In the same way, we can use any of the other four approaches (*Cet-cvx, Cet-loc, Cet-loc-cvx, Seol*) to initialize our model, however none of them exhibits a significant advantage (see Figure 15); hence we proceed with the above two initialization strategies.

To evaluate data fidelity, we will use a mean squared error (MSE):

$$MSE(\widehat{\mathbf{b}}, \tilde{\mathbf{b}}) = \frac{1}{n} \sum_{i=1}^{n} (\hat{b}_i - \tilde{b}_i)^2, \qquad (12)$$

where $\widehat{\mathbf{b}}$ 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` [55], hence all the methods are evaluated on the same level of rig complexity. Since the animation data used in this paper is created manually, we have available ground-truth weight vectors, but we are not interested in matching the weights, only the meshes. While this is the metric of main importance to us, we also want a solution to be as sparse as possible while keeping the mesh fidelity high (i.e. MSE in (12) low). An appropriate metric for this is the cardinality of a predicted weight vector, i.e., a 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 it will still count when measuring cardinality. For this reason we include an additional indicator of sparsity — L_1 norm of the solution vector.

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.

We chose `MetaHuman` characters since they alighn well with the assumptions of our model and are increasingly popular choice for building realistic face animation [52], [53]

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.

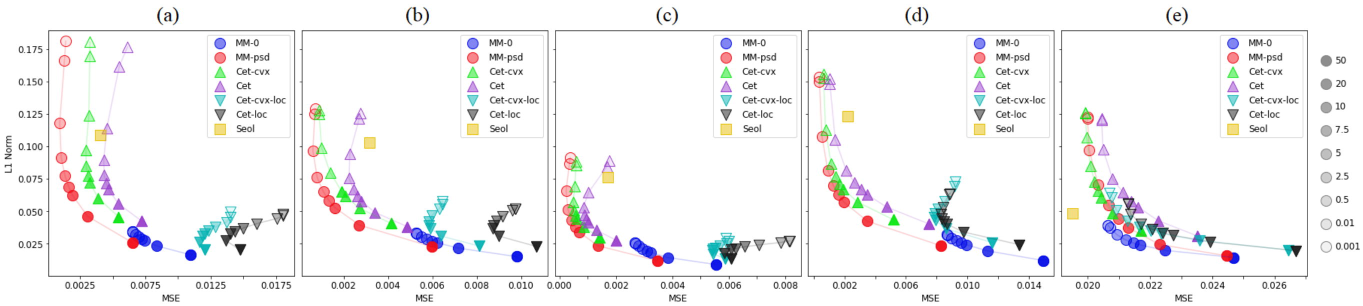

Fig. 6. Trade-off between mesh error (MSE) and $l1$ norm of the estimated solutions for different methods and varying values of regularization parameter α. (a) *Omar*, (b) *Danielle*, (c) *Myles*, (d) *Char 4*, (e) *Char 5*.

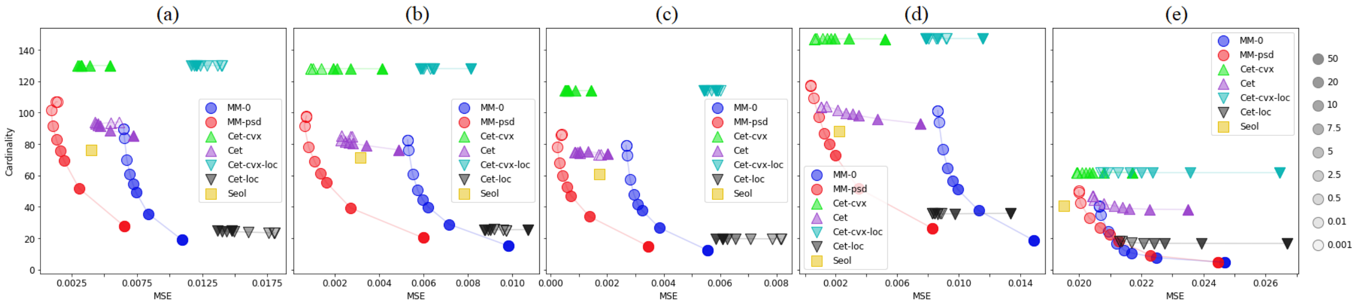

Fig. 7. Trade-off between mesh error (MSE) and cardinality of the estimated solutions for different methods and varying values of regularization parameter α. (a) *Omar*, (b) *Danielle*, (c) *Myles*, (d) *Char 4*, (e) *Char 5*.

Our method, with two initialization approaches, as well as the other benchmark models, are tested with a wide range of regularization parameter values $\alpha \in \{0.001, 0.01, 0.5, 2.5, 5, 7.5, 10, 20, 50\}$. The desired solution should have high data fidelity while keeping the number of activated components low, hence we look at the trade-off curves between MSE (mesh error) and cardinality of the weight vectors in Figure 7. As one can see, our approach with pseudoinverse initialization (*MM-psd*) yields a curve that is always below the others, except in the case of *Char 5*, where *MM-0* and *Seol* are compatible with it. Similar behavior is observed if we consider a trade-off with MSE and $L1$ norm (Figure 6). To compare the results in more detail, we pick an optimal value of α, using the elbow method, for each method and each dataset.

Figure 8 gives the values of the three metrics for *Omar*, as well as the average computational time per frame. Our method (*MM-psd*) shows the best performance in terms of data fidelity while keeping both the cardinality and the volume of a weight vector reasonably low. The only aspect where other methods exhibit better performance is execution speed. However, since this method targets the production of movie and game animation, it is not restricted to real-time computations, and performance speed in our results (order of a dozen of seconds) is feasible. Note that here we implemented the algorithm sequentially, but due to the construction, *inner iterations* could be implemented in parallel to further reduce the execution time.

In Figure 1 we see an example frame for character *Omar*, with meshes and activation weights. Although *Cet* provides a close fit, one can see that our method *MM-psd* better fits details below the lower lip as well as cheek wrinkles. Localized methods *Cet-loc* and *Cet-loc-cvx* in general give a poor fit,

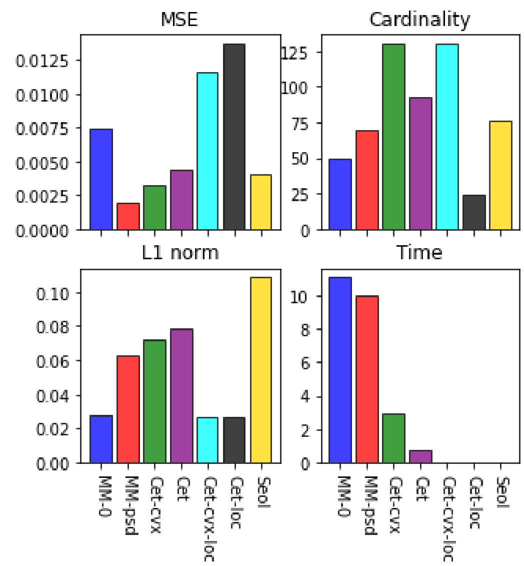

Fig. 8. Values of the three metrics (mesh MSE, weights cardinality and weights $L1$ norm) and execution time (in seconds) for *Omar*. Execution time for the last three methods is not visible, and it is $9.65e^{-05}$, $1.14e^{-04}$ and 0.02 respectively.

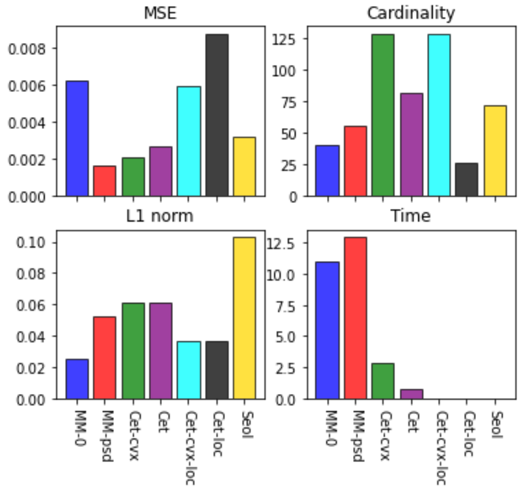

Fig. 9. Values of the three metrics (mesh MSE, weights cardinality and weights $L1$ norm) and execution time (in seconds) for *Danielle*. Execution time for the last three methods is not visible, and it is $8.32e^{-05}$, $1.00e^{-04}$ and 0.02 respectively.

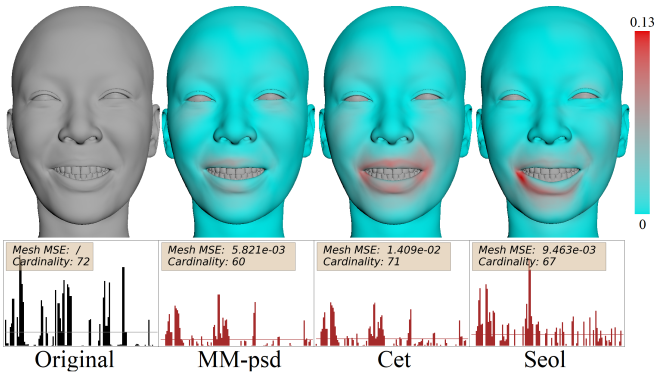

Fig. 10. Example frame prediction for *Danielle*. The top row shows obtained meshes, while the bottom represents corresponding activations of the controller weights. Red tones in the meshes indicate higher error of the fit, according the colorbar on the right. The average weight activation of each solution is indicated with a horizontal line. The average mesh error and cardinality of the solution are given in a textbox.

and the obtained expression is semantically different from the original, hence we will dismiss them in the analysis of the following results. Similar holds for our method with zero initialization (*MM-0*) which is why we excluded it. Finally, *Cet-cvx* yields meshes that are almost indistinguishable from *Cet* (while it is slower to compute), hence we will omit it as well, and for the other characters we show only meshes obtained by *MM-psd, Cet* and *Seol*.

Metric values for *Danielle* are given in Figure 9 and an example frame in Figure 10. Both *Cet* and *Seol* give a mesh with lips not wide enough, while *Seol* additionally produces much higher number of non-zero weights. For *Myles*, mesh error of *Cet* is comparable with our, but our solution is significantly more sparse. Figure 11 shows clear errors in the lower lip for our method and *Cet*, but on the other side we can see that *Seol* produces slight red tones over the entire face, and in the end it gives a higher average MSE. In the case of *Char 4* the conclusions are similar like for *Myles* (Figure 13). Finally, for *Char 5* all the approaches have similar mesh

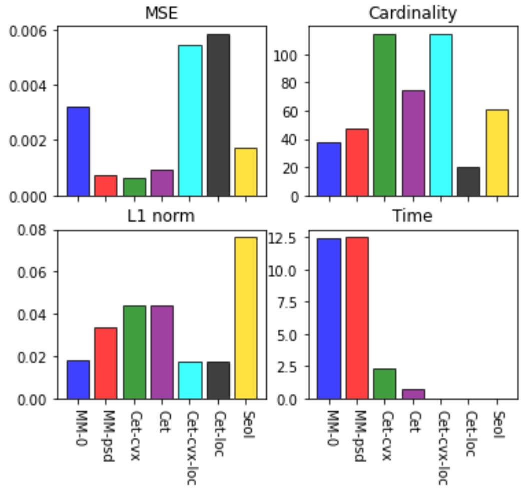

Fig. 11. Values of the three metrics (mesh MSE, weights cardinality and weights $L1$ norm) and execution time (in seconds) for *Myles*. Execution time for the last three methods is not visible, and it is $8.07e^{-05}$, $9.24e^{-05}$ and 0.02 respectively.

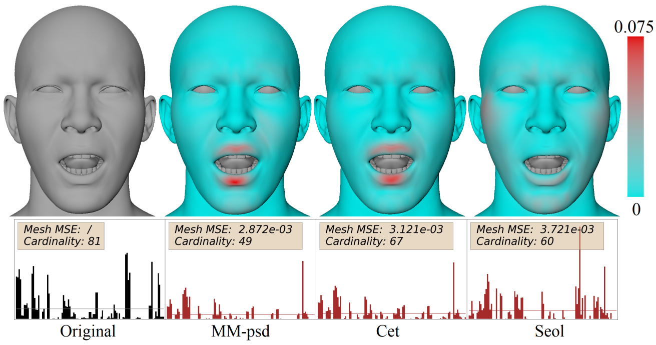

Fig. 12. Example frame prediction for *Myles*. The top row shows obtained meshes, while the bottom represents corresponding activations of the controller weights. Red tones in the meshes indicate higher error of the fit, according the colorbar on the right. The average weight activation of each solution is indicated with a horizontal line. The average mesh error and cardinality of the solution are given in a textbox.

error (Figure 14), hence we cannot claim the superiority of our method here. This comes as no surprise since we mentioned that this face model has a different structure, with many non-blendshape components, and our algorithm is targeting detailed and accurate blendshape rigs. However, this shows that even with relaxed assumptions about the face model, our method is comparable to state-of-the-art solutions.

In Figure 15 we see the trade-off curves for different initialization vectors using our majorization minimization algorithm. None of the choices shows a superior performance compared to the two initializations that we considered initially (*MM-0, MM-psd*), hence we have no motivation to include any of these additional strategies in the overall discussion.

VI. Conclusion

The method proposed in this paper applies a majorization minimization paradigm in order to allow incorporation of the quadratic corrective terms of the blendshape models when solving the inverse rig problem. It gives better fit in the details

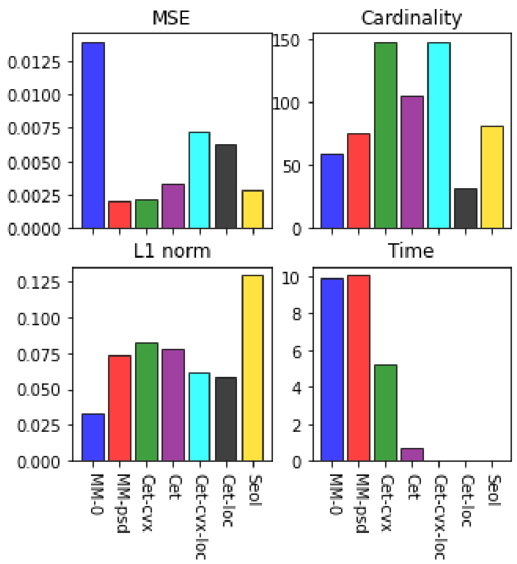

Fig. 13. Values of the three metrics (mesh MSE, weights cardinality and weights $L1$ norm) and execution time (in seconds) for *Char 4*. Execution time for the last three methods is not visible, and it is $1.02e^{-04}$, $1.22e^{-04}$ and 0.02 respectively.

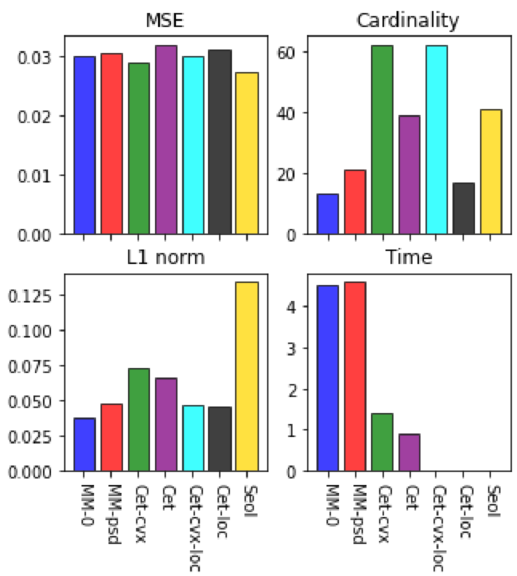

Fig. 14. Values of the three metrics (mesh MSE, weights cardinality and weights $L1$ norm) and execution time (in seconds) for *Char 5*. Execution time for the last three methods is not visible, and it is $6.94e^{-05}$, $4.65e^{-05}$ and 0.02 respectively.

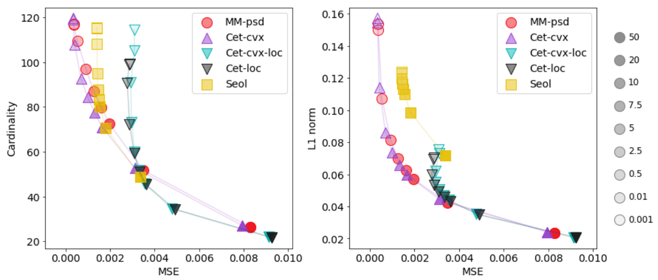

Fig. 15. Trade-off between mesh error and cardinality for our method using different initialization vectors.

of the face mesh while not increasing the cardinality of the weight vector, hence the model 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 the future work we will address this to additionally reduce the execution time. Another aspect that we will address in the future research is to include an additional step of face segmentation, that might lead to distributed model and possibly even higher precision in fitting the fine details of the face mesh.

ACKNOWLEDGEMENTS

The authors thank John Doe for the collaboration of preparing this paper. This Project was financed by the CNPq (xxyyzz process).

REFERENCES

[1] O. Cetinaslan, V. Orvalho, "Sketching Manipulators for Localized Blendshape Editing", *Graphical Models*, vol. 108, p. 101059, 2020.

[2] Y. Seol, J. Seo, P. H. Kim, J. P. Lewis, J. Noh, "Artist Friendly Facial Animation Retargeting", *in Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, Association for Computing Machinery, New York, NY, USA, 2011, doi:10.1145/2024156.2024196, URL: `https://doi.org/10.1145/2024156.2024196`.

[3] S. Racković, C. Soares, D. Jakovetić, Z. Desnica, "A Majorization-Minimization Based Method for Nonconvex Inverse Rig Problems in Facial Animation: Algorithm Derivation", , 2022, URL: `https://gitfront.io/r/user-6651490/5d38163c0ca72a26cea5b70028bf84a18d793392/manuscripts/`.

[4] F. H. Pighin, J. Hecker, D. Lischinski, R. Szeliski, D. Salesin, "Synthesizing realistic facial expressions from photographs", *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998.

[5] B. Choe, H.-S. Ko, "Analysis and synthesis of facial expressions with hand-generated muscle actuation basis", *in ACM SIGGRAPH 2006 Courses*, pp. 21–es, 2006.

[6] B. Choe, H. Lee, H.-S. Ko, "Performance-driven muscle-based facial animation", *The Journal of Visualization and Computer Animation*, vol. 12, no. 2, pp. 67–79, 2001.

[7] M. D. Smet, L. V. Gool, "Optimal regions for linear model-based 3d face reconstruction", *in Asian conference on computer vision*, pp. 276–289, Springer, 2010.

[8] T. Neumann, K. Varanasi, S. Wenger, M. Wacker, M. Magnor, C. Theobalt, "Sparse localized deformation components", *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, pp. 1–10, 2013.

[9] B. Chaudhuri, N. Vesdapunt, L. Shapiro, B. Wang, "Personalized face modeling for improved face reconstruction and motion retargeting", *in European Conference on Computer Vision*, pp. 142–160, Springer, 2020.

[10] M. Wang, D. Bradley, S. Zafeiriou, T. Beeler, "Facial Expression Synthesis using a Global-Local Multilinear Framework", *in Computer Graphics Forum*, vol. 39, pp. 235–245, Wiley Online Library, 2020.

[11] H. Li, T. Weise, M. Pauly, "Example-based facial rigging", *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, pp. 1–6, 2010.

[12] H. Li, J. Yu, Y. Ye, C. Bregler, "Realtime facial animation with on-the-fly correctives.", *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 42–1, 2013.

[13] R. B. i. Ribera, E. Zell, J. P. Lewis, J. Noh, M. Botsch, "Facial Retargeting with Automatic Range of Motion Alignment", *ACM Trans Graph*, vol. 36, no. 4, jul 2017, doi:10.1145/3072959.3073674, URL: `https://doi.org/10.1145/3072959.3073674`.

[14] J. Zhang, K. Chen, J. Zheng, "Facial expression retargeting from human to avatar made easy", *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 2, pp. 1274–1287, 2020.

[15] J. H. Han, J.-I. Kim, H. Kim, J. W. Suh, "Generate Individually Optimized Blendshapes", *in 2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 114–120, 2021, doi:10.1109/BigComp51126.2021.00030.

[16] Z. Deng, P.-Y. Chiang, P. Fox, U. Neumann, "Animating Blendshape Faces by Cross-Mapping Motion Capture Data", *in Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, I3D '06, p. 43–48, Association for Computing Machinery, New York, NY, USA, 2006, doi:10.1145/1111411.1111419, URL: `https://doi.org/10.1145/1111411.1111419`.

[17] Y. Seol, J. P. Lewis, "Tuning Facial Animation in a Mocap Pipeline", *in ACM SIGGRAPH 2014 Talks*, SIGGRAPH '14, Association for Computing Machinery, New York, NY, USA, 2014, doi:10.1145/2614106.2614108, URL: `https://doi.org/10.1145/2614106.2614108`.

[18] E. Sifakis, I. Neverov, R. Fedkiw, "Automatic Determination of Facial Muscle Activations from Sparse Motion Capture Marker Data", *in ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, p. 417–425, Association for Computing Machinery, New York, NY, USA, 2005, URL: `https://doi.org/10.1145/1186822.1073208`.

[19] K. Hirose, T. Higuchi, "Creating facial animation of characters via MoCap data", *Journal of Applied Statistics*, vol. 39, no. 12, pp. 2583–2597, 2012, doi:10.1080/02664763.2012.724391, URL: `https://doi.org/10.1080/02664763.2012.724391`, `https://doi.org/10.1080/02664763.2012.724391`.

[20] Y. Seol, J. Lewis, J. Seo, B. Choi, K. Anjyo, J. Noh, "Spacetime Expression Cloning for Blendshapes", *ACM Trans Graph*, vol. 31, no. 2, apr 2012, doi:10.1145/2159516.2159519, URL: `https://doi.org/10.1145/2159516.2159519`.

[21] C. Ouzounis, A. Kilias, C. Mousas, "Kernel Projection of Latent Structures Regression for Facial Animation Retargeting", *in Proceedings of the 13th Workshop on Virtual Reality Interactions and Physical Simulations*, VRIPHYS '17, p. 59–65, Eurographics Association, Goslar, DEU, 2017, doi:10.2312/vriphys.20171084, URL: `https://doi.org/10.2312/vriphys.20171084`.

[22] C. Reverdy, S. Gibet, C. Larboulette, "Optimal Marker Set for Motion Capture of Dynamical Facial Expressions", *in Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, MIG '15, p. 31–36, Association for Computing Machinery, New York, NY, USA, 2015, doi:10.1145/2822013.2822042, URL: `https://doi.org/10.1145/2822013.2822042`.

[23] D. Bradley, W. Heidrich, T. Popa, A. Sheffer, "High Resolution Passive Facial Performance Capture", *ACM Trans Graph*, vol. 29, no. 4, jul 2010, doi:10.1145/1778765.1778778, URL: `https://doi.org/10.1145/1778765.1778778`.

[24] T. Beeler, F. Hahn, D. Bradley, B. Bickel, P. Beardsley, C. Gotsman, R. W. Sumner, M. Gross, "High-Quality Passive Facial Performance Capture Using Anchor Frames", *ACM Trans Graph*, vol. 30, no. 4, jul 2011, doi:10.1145/2010324.1964970, URL: `https://doi.org/10.1145/2010324.1964970`.

[25] J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, C. Theobalt, "Real-Time Expression Transfer for Facial Reenactment", *ACM Trans Graph*, vol. 34, no. 6, oct 2015, doi:10.1145/2816795.2818056, URL: `https://doi.org/10.1145/2816795.2818056`.

[26] D. Holden, J. Saito, T. Komura, "Learning Inverse Rig Mappings by Nonlinear Regression", *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 3, pp. 1167–1178, 2016.

[27] S. W. Bailey, D. Omens, P. Dilorenzo, J. F. O'Brien, "Fast and Deep Facial Deformations", *ACM Trans Graph*, vol. 39, no. 4, jul 2020, doi:10.1145/3386569.3392397, URL: `https://doi.org/10.1145/3386569.3392397`.

[28] S. L. Song, W. Shi, M. Reed, "Accurate Face Rig Approximation with Deep Differential Subspace Reconstruction", *ACM Trans Graph*, 2020.

[29] S. Kim, S. Jung, K. Seo, R. B. i Ribera, J. Noh, "Deep Learning-Based Unsupervised Human Facial Retargeting", *Computer Graphics Forum*, vol. 40, 2021.

[30] J. Song, B. Choi, Y. Seol, J. yong Noh, "Characteristic facial retargeting", *Computer Animation and Virtual Worlds*, vol. 22, 2011.

[31] D. Holden, J. Saito, T. Komura, "Learning an inverse rig mapping for character animation", *in Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 165–173, 2015.

[32] W.-W. Feng, B.-U. Kim, Y. Yu, "Real-Time Data Driven Deformation Using Kernel Canonical Correlation Analysis", *ACM Trans Graph*, vol. 27, no. 3, p. 1–9, aug 2008, doi:10.1145/1360612.1360690, URL: `https://doi.org/10.1145/1360612.1360690`.

[33] H. Yu, H. Liu, "Regression-based facial expression optimization", *IEEE Transactions on Human-Machine Systems*, vol. 44, no. 3, pp. 386–394, 2014.

[34] S. Bouaziz, Y. Wang, M. Pauly, "Online Modeling for Realtime Facial Animation", vol. 32, no. 4, jul 2013, doi:10.1145/2461912.2461976, URL: `https://doi.org/10.1145/2461912.2461976`.

[35] C. O. Çetinaslan, *Position Manipulation Techniques for Facial Animation*, Tese de Doutorado, Faculdade de Ciencias da Universidade do Porto, 2016.

[36] J. Song, R. Blanco i Ribera, K. Cho, M. You, J. P. Lewis, B. Choi, J. Noh, "Sparse Rig Parameter Optimization for Character Animation", *in Computer Graphics Forum*, vol. 36, pp. 85–94, Wiley Online Library, 2017.

[37] K. Anjyo, J. Lewis, "Direct Manipulation Blendshapes", *IEEE Computer Graphics and Applications*, vol. 30, no. 04, pp. 42–50, 2010.

[38] J. Seo, G. Irving, J. P. Lewis, J. Noh, "Compression and Direct Manipulation of Complex Blendshape Models", *ACM Trans Graph*, vol. 30, no. 6, p. 1–10, dec 2011, doi:10.1145/2070781.2024198, URL: `https://doi.org/10.1145/2070781.2024198`.

[39] O. Cetinaslan, V. Orvalho, "Stabilized blendshape editing using localized Jacobian transpose descent", *Graphical Models*, vol. 112, p. 101091, 2020.

[40] K.-Y. Liu, W.-C. Ma, J. Lee, C.-C. Wang, C.-F. Chang, "Localized Optimization for Mocap-Driven Blendshapes", *in SCA 2010*, 2010.

[41] K.-G. Na, M.-R. Jung, "Local shape blending using coherent weighted regions", *The Visual Computer*, vol. 27, no. 6-8, p. 575, 2011.

[42] J. R. Tena, F. De la Torre, I. Matthews, "Interactive Region-Based Linear 3D Face Models", *in ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, Association for Computing Machinery, New York, NY, USA, 2011, URL: `https://doi.org/10.1145/1964921.1964971`.

[43] M. Fratarcangeli, D. Bradley, A. Gruber, G. Zoss, T. Beeler, "Fast Nonlinear Least Squares Optimization of Large-Scale Semi-Sparse Problems", *Computer Graphics Forum*, vol. 39, 2020.

[44] D. L. James, C. D. Twigg, "Skinning mesh animations", *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 399–407, 2005.

[45] P. Joshi, W. C. Tien, M. Desbrun, F. Pighin, "Learning controls for blend shape based realistic facial animation", *in ACM SIGGRAPH 2006 Courses*, pp. 17–es, 2006.

[46] M. Romeo, S. Schvartzman, "Data-Driven Facial Simulation", *in Computer Graphics Forum*, vol. 39, pp. 513–526, Wiley Online Library, 2020.

[47] S. Racković, C. Soares, D. Jakovetić, Z. Desnica, R. Ljubobratović, "Clustering of the Blendshape Facial Model", *in 2021 29th European Signal Processing Conference (EUSIPCO)*, pp. 1556–1560, 2021, doi: 10.23919/EUSIPCO54536.2021.9616061.

[48] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. H. Pighin, Z. Deng, "Practice and Theory of Blendshape Facial Models.", *Eurographics (State of the Art Reports)*, vol. 1, no. 8, p. 2, 2014.

[49] M. Schmidt, "Least squares optimization with L1-norm regularization", *CS542B Project Report*, vol. 504, pp. 195–221, 2005.

[50] E. D. Schifano, R. L. Strawderman, M. T. Wells, "Majorization-Minimization algorithms for nonsmoothly penalized objective functions", *Electronic Journal of Statistics*, vol. 4, pp. 1258–1299, 2010.

[51] Y. Sun, P. Babu, D. P. Palomar, "Majorization-Minimization algorithms in signal processing, communications, and machine learning", *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 794–816, 2016.

[52] Z. Fang, L. Cai, G. Wang, "MetaHuman Creator The starting point of the metaverse", *in 2021 International Symposium on Computer Technology and Information Science (ISCTIS)*, pp. 154–157, IEEE, 2021.

[53] J. R. Siddiqui, "FExGAN-Meta: Facial Expression Generation with Meta Humans", , 2022.

[54] S. Diamond, S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization", *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[55] Autodesk, INC., "Maya", 2019, 2019-01-15, URL: `https:/autodesk.com/maya`.