DOI: 10.1111/cgf.13925
EUROGRAPHICS 2020 / U. Assarsson and D. Panozzo
(Guest Editors)
Volume 39 (2020), Number 2

Expression Packing: As-Few-As-Possible Training Expressions for Blendshape Transfer

E. Carrigan[1], E. Zell[1], C. Guiard[2] and R. McDonnell[1]

[1]Trinity College Dublin [2]EISKO

Abstract

To simplify and accelerate the creation of blendshape rigs, using a template rig is a common procedure, especially during the creation of digital doubles. Blendshape transfer methods facilitate copy and paste functionality of the blendshapes from the template model to the digital double. However, for adequate personalization, such methods require a set of scanned training expressions of the original actor. So far, the semantics of the facial expressions to scan have been defined manually. In contrast, we formulate the semantics of the facial expressions as an integer optimization of the blendshape weights. By combining different blendshapes of the template model, our method creates facial expressions that serve as semantic references during scanning. Our method guarantees to compute as-few-as-possible training expressions with minimal overlap of activated blendshapes. If the number of training expressions is limited, blendshapes are selected based on their power to personalize the resulting blendshapes compared to generic blendshape transfer methods.

CCS Concepts

• ***Theory of computation*** → ***Packing and covering problems;*** • ***Computing methodologies*** → ***Animation;***

1. Introduction

Creating high-quality, production-ready animation rigs for individual characters is a time-consuming task which is a major bottleneck in current facial animation pipelines, where blendshape interpolation is still the method of choice for real-time [Sey16] and offline animation [Sey19]. Aiming for high quality, it became common to acquire 3d scans for digital doubles, while high-resolution models of fictional characters are sculpted in 3d. The resulting 3d models and expressions define the shape down to the pore level, but vary with regard to the vertex count and are therefore lacking the required one-to-one vertex correspondence between blendshapes. In order to ensure equal numbers of vertices and consistent connectivity across expressions, the 3d meshes are retopologized, in general at lower resolution. Academic work largely automated this process by registering a template blendshape model non-rigidly towards the 3d scan or sculpted model (e.g., [IBP15, FNH*17, LBB*17]). Within these frameworks, the template model is consistently deformed until it accurately approximates the shape of the 3d scan or sculpted model. To simplify the creation of blendshapes further, several algorithms have been proposed that transfer existing generic blendshapes from a template model to new characters [SP04] or personalize the generic blendshapes based on training expressions [LWP10, SML16]. So far, the semantics of these training expressions, whether it should be a smile, frown or any other expression, have been defined manually.

In general, blendshape transfer methods offer a trade-off between fast generation of plausible (but not necessarily artefact-free) blendshapes and creation of accurately-personalized blendshapes, at the cost of requiring many training expressions. While the two extremes are well defined (either no training expressions or one training expression for each blendshape) no obvious solution exists for an optimal set of training expressions. By optimal, we mean that we aim to satisfy the following four goals:

1. The number of training expressions should be as-few-as-possible to reduce the number of expressions to scan or sculpt.
2. To ensure accurate extraction of personalized blendshapes, no overlap of blendshapes should exist in training expressions. (e.g., instead of combining two mouth blendshapes, a combination of an eye and a mouth blendshape should be preferred).
3. If the number of training expressions is limited, blendshapes that would be problematic to transfer without examples, should be prioritized.
4. Training expressions should be poseable, meaning that a human face should be able to reproduce them.

After reviewing recent blendshape creation pipelines [FNH*17], we observe that the template blendshape model used in blendshape transfer methods is actually available before individual expressions are scanned or sculpted. Because the blendshapes of the template and the final, personalized rig will be similar, we can combine

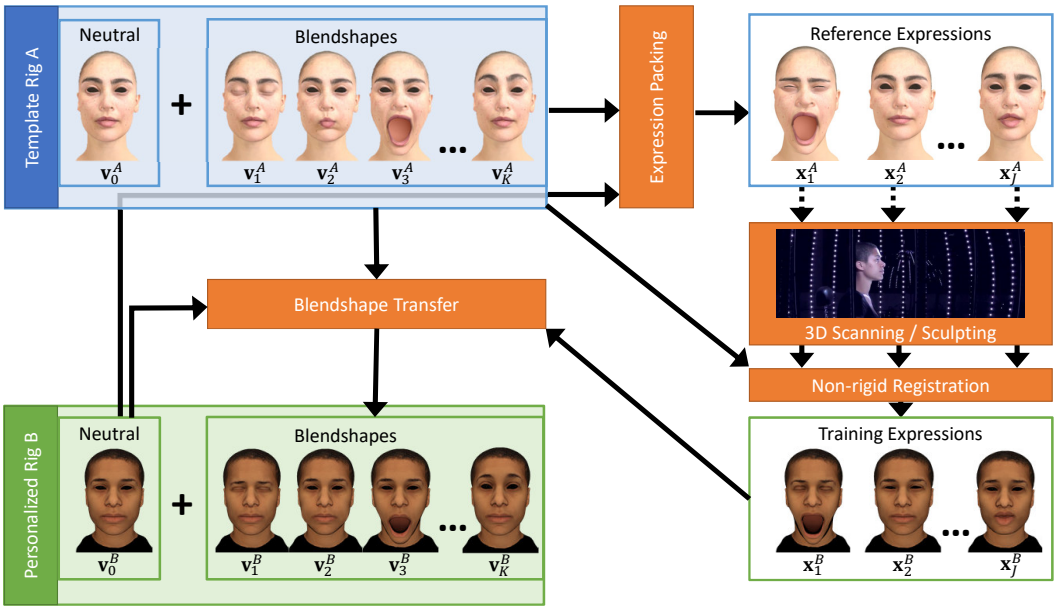

Figure 1: *Illustration of our expression packing algorithm as part of the digital double pipeline. We run our expression packing algorithm to create as-few-as-possible reference expressions, which help to guide an actor during a scanning session. Using non-rigid registration, e.g. [IBP15, FNH*17, LBB*17], the mesh topology of the template rig is transferred to the 3d scans. The resulting training expressions are optimal for the blendshape transfer algorithm, since they lead to best results with as-few-as-possible expressions. Finally, the blendshapes are computed for the personalized rig.*

several non-interfering blendshapes to create complex expressions and use them during scanning as semantic references for the actual training expressions (Figure 1). To our knowledge, we present the first numerical optimization method that automatically combines blendshapes to form optimal expressions. In addition, results of our optimization largely satisfy the above mentioned criteria. We pre-compute a minimal set of reference expressions, ordered by their power to improve the overall results of the blendshape transfer operation. Considering only the first reference expressions of our minimal set will improve blendshape transfer methods as-much-as-possible for an incomplete set of training expressions. This is an often encountered practical case, for example if capturing-time is short due to the availability of celebrities, or the budget limits the number of training expressions that can be post-processed. An entire run of our optimization returns the smallest set of reference expressions for each blendshape and this set is significantly smaller than the number of blendshapes of the template rig. Finally, our method is generic as it is suitable for any blendshape basis. A reference implementation is published on GitHub to facilitate replication and comparison[†].

2. Related Work

Facial animation based on linear interpolation of blendshapes [LAR*14] remains highly popular due to the numerical simplicity and intuition both in practical applications as well as in research [PHL*06, Osi07, DN08, OBP*12]. In addition, personalized 3d blendshape models form the basis for various markerless facial capturing methods [WBLP11,BWP13,LYYB13,CHZ14, TZN*15], where the objective is to fit a linear face model to an image [BV*99]. The main advantage of blendshapes over principal component analysis (PCA), independent component analysis and linear discriminant analysis etc. is the semantic encoding of expressions. Despite many advantageous mathematical properties, PCA eigenvectors do not guarantee semantically meaningful expressions, especially for eigenvectors associated with the later eigenvalues (10th and later) [LAR*14].

Because the manual creation of a blendshape model is a time-consuming task [Osi07], several methods have been developed to facilitate the transfer of blendshapes to new characters. In expression cloning [NN01], the direction and magnitude of vertex-displacements is adjusted separately. Other authors proposed radial basis functions (RBF) [Pan03, FSF07, OZS08] or deep learning [GYQ*18] to relax or completely remove the need for dense correspondences between meshes. Sumner et al. [SP04] couple local rotation and scaling of triangles by transferring deformation gradients. The method was later improved by adding semantic constraints [Sai13] or physical constraints like collision detection [IKNDP16]. Deformation transfer [SP04] is also closely related to various non-linear shape-interpolation methods, e.g. [BVGP09].

Li et al. [LWP10] proposed to personalize the generic blendshape transfer method by providing a small set of training expressions in combination with approximated blendshape weights. Recently, an extended method has been applied in VFX produc-

† https://github.com/Fiquem/Expression-Packing

tions, where all blendshapes are covered by training expressions [SML16]. After combining several blendshapes in one expression, the number of training expressions is significantly lower compared to the number of blendshapes. In general, combining non-overlapping blendshapes in training expressions is preferred for easier separation [CFA*16]. Methods for creating high-quality digital doubles either from photographs [PHL*06, HSW*17] or 3D scans [ZSCS08, GZC*16, FNH*17] is a recurrent topic in computer graphics. Personalized blendshapes can also be estimated using multi-linear interpolations based on a big database of different people with ideally, but not necessarily, semantically equivalent expressions. Suitable databases are Facewarehouse [CWZ*14], FLAME [LBB*17] or internal datasets [SL14, HSW*17]. Unfortunately, the level of detail of public datasets is dramatically lower compared to high-resolution rigs using photogrammetry [FNH*17] – a few thousand vertices vs. pore-deep reconstruction level.

Automatic transfer of blendshapes, either using radial basis functions or deformation transfer is also relevant within the context of facial animation retargeting [Pan03, FSF07, OZS08, SILN11, SLS*12]. Within this domain, creating two semantically equivalent blendshape models allows simple copying of blendshape weights for animation. Improving accuracy of the sparse blendshapes rigs for retargeting with examples [LMX*08], or based on automatically detected fuzzy correspondences [RZL*17] significantly improved animation retargeting. It should be noted that some earlier work on facial animation retargeting was named as example-based facial animation [PKC*03, PCNS05, SCSN11, BP14]. However, instead of computing blendshapes, the aim of this work is to transfer the actual animation, once the blendshapes are known. Starting from semantically equivalent expression pairs of two characters, a direct mapping is learned between the parameter spaces of the rigs.

While many methods rely on training expressions to improve the accuracy of blendshape transfer methods, the question of which blendshapes should be corrected by a training expression has barely been addressed. So far, training expressions are defined manually and refined over time using a trial and error process. One notable exception is the perceptual evaluation by Carrigan et al. [CHMA18], however their analysis is limited to action units as defined by FACS. In practice, the number of blendshapes vary between 30 and over 200, and while most are inspired by the FACS system [EF78], it is less common to encounter exact reproduction of all action units. In contrast, we present a method that automatically identifies blendshapes that are difficult to transfer, is suitable for any blendshape basis, and by design preserves all benefits of example based methods [LWP10, SML16].

We formulate our optimization problem for computing the semantics of the training expressions as a linear integer programming problem. Within the computer graphics field, mixed integer programming was previously applied to shape segmentation [HKG11], correspondence search between unregistered shapes [VLR*17] and to optimize layouts [FDH*15, WFLW18]. More details on the application of mixed integer programming in computer graphics can be found in a recent course [Won18].

3. Blendshape Transfer

An example-based blendshape transfer method requires a template blendshape rig $\mathcal{A}$, consisting of triangle meshes posing a neutral expression and K blendshapes. All meshes are of identical connectivity and N vertices (after non-rigid mesh registration [FNH*17]). We denote the neutral expression and all blendshapes as the set $\mathcal{A} = \{\mathbf{v}_0^A, \mathbf{v}_1^A, \dots, \mathbf{v}_K^A\}$, where the vertex positions $\mathbf{v}_k^n$ are stacked in a single vector $\mathbf{v}_k = (\mathbf{v}_k^n, \dots, \mathbf{v}_k^N)^T$ of size $3N$ due to the coupling of xyz-coordinates. Delta-blendshapes are defined as: $\delta\mathbf{v}_k = \mathbf{v}_k - \mathbf{v}_0$. In addition to the template blendshape model, a target model exists of different identity $\mathbf{v}_0^B$. For the target model $\mathcal{B}$, J training expressions $\mathbf{x}_j^B$ are provided ($\mathcal{B} = \{\mathbf{x}_1^B, \dots, \mathbf{x}_J^B\}$) with the same number of vertices and connectivity as $\mathcal{A}$. For each training expression, the (approximate) blendshape weights w_k^j are known and the goal of the blendshape transfer function is to compute the missing personalized blendshapes $\{\mathbf{v}_1^B, \dots, \mathbf{v}_K^B\}$ of $\mathcal{B}$, satisfying the equation

$$\mathbf{x}_j^B(\mathbf{v}_1^B, \dots, \mathbf{v}_K^B) = \mathbf{v}_0^B + \sum_{k=1}^{K} w_k^j \left(\mathbf{v}_k^B - \mathbf{v}_0^B\right) \tag{1}$$

We assume that the two blendshapes $\mathbf{v}_k^A$ and $\mathbf{v}_k^B$ are semantically equivalent expressions of $\mathcal{A}$ and $\mathcal{B}$ and of identical connectivity. In our work, we focus on finding the semantically equivalent expressions $\mathbf{x}_j^A$ that serve as a reference for creating the training expressions $\mathbf{x}_j^B$, either by modelling the expressions manually or by posing the expression for a 3d-scan. Our main intention is to obtain the semantics for as-few-as-possible training expressions J, and obtain the most accurate unknown blendshapes $\{\mathbf{v}_1^B, \dots, \mathbf{v}_K^B\}$ as a result of the example-based blendshape transfer method. A high-level overview of this method can be seen in Figure 1. For greater clarity and convenience we summarize our notation in Table 1.

4. Optimal Reference Expressions

In the following, we will derive step-by-step our optimization method for pre-computing the reference expressions $\mathbf{x}_j^A$ for any blendshape basis. To simplify notation, we largely omit the indices A and B within this section. Variables without an index refer to the blendshape model $\mathcal{A}$, e.g. $\mathbf{v}_k \equiv \mathbf{v}_k^A$. We first introduce the concept of binary blendshapes (Section 4.1), which is a fundamental conceptual basis for our optimization framework (Section 4.2). Afterwards, we discuss important modifications to obtain numerically optimal as well as plausible expressions (Sections 4.3, 4.4). We close this section with a discussion on how to solve the optimization function and the benefits of different solutions (Section 4.5).

4.1. Binary Blendshapes

Example-based methods estimate personalized individual blendshapes from a training expression. This problem is ill-posed when an expression consists of overlapping blendshapes (e.g., if an expression is composed of a smile and a mouth-open blendshape). In contrast, computing individual blendshapes is easy from a training expression consisting of, for example, an eye-closing and a mouth opening blendshape as there is no overlap. To reduce ambiguity within training expressions, we search for a metric that allows a balance between combining as many blendshapes as possible in one

Variable	Description
j, J	index / absolute number of examples
n, N	index / absolute number of vertices
t, T	index / absolute number of triangles
k, K / m, M	index / absolute number of blendshapes
$\mathcal{A}, \mathcal{B}$	blendshape rigs
$\mathbf{v}_k^n$	position of vertex n of blendshape k
$\mathbf{v}_0, \mathbf{v}_k$	neutral expressions and blendshape k
$\delta\mathbf{v}_k$	delta-blendshape k, with: $\delta\mathbf{v}_k = \mathbf{v}_k$-$\mathbf{v}_0$
$\delta\mathbf{v}_k^n$	displacement of vertex n and blendshape k
w_n	weight of blendshape n
$\mathbf{x}_j$	example expression
$\mathbf{b}_k$	binary blendshape n
b_k^n	value for vertex n in $\mathbf{b}_k$, $b_k^n \in \{0,1\}$
λ	blendshape importance weight
s	distortion metric between two triangles
d	euclidean distance between two vertices
$\mathbf{u}, \mathbf{v}, \mathbf{w}$	vertex positions of a triangle $\mathbf{uvw}$
$(w^{\llcorner}, w^{\ulcorner})$	blendshape weights of a symmetric pair

Table 1: *Notation overview of the most relevant variables.*

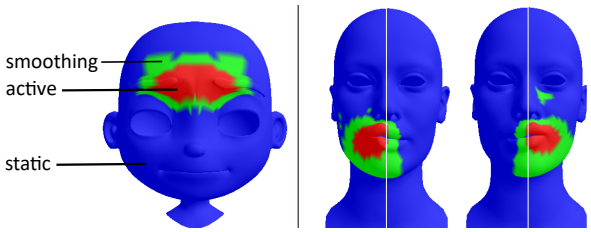

Figure 2: *Illustration of the different types of vertices for one blendshape: static (blue), smoothing (green) and active (red). The blendshape on the left is self-symmetric, the two blendshapes on the right are symmetric, but have overlapping active vertices.*

training expression and preventing too strong an overlap between blendshapes within the training expressions (Goal 2, Section 1).

For this purpose, we introduce a new concept of binary blendshapes (eq. 2). Delta-blendshapes $\delta\mathbf{v}_k$ define vertex displacement with respect to the neutral expression $\mathbf{v}_0$, and binary blendshapes $\mathbf{b}_k$ will contain the information of the location of relevant deformation within the blendshape. In the following, vertices of blendshape k with zero displacement ($\delta\mathbf{v}_k^n = 0$) will be called *static*. In addition, vertices with a displacement bigger than zero will be divided into two stages: *active* and *smoothing* vertices, depending whether their displacement is strong and important for recognizing the semantic meaning or whether the deformation mainly exists to maintain a smooth deformation. See Figure 2 for an illustration. It is worth mentioning that the differentiation between static and non-static vertices is a common part of example-based blendshape transfer methods, either in the form of soft constraints [LWP10] - remain similar to the original blendshape, or hard constraints [SML16] - static vertices remain static.

$$b_k^n = \begin{cases} 0 & \text{if } \|\delta\mathbf{v}_k^n\| < \mu \max_{n \in N} \|\delta\mathbf{v}_k^n\|, \quad 0.25 \leq \mu \leq 0.4 \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

The information of whether a vertex is active or non-active (static or smoothing) is saved as a binary value $b_k^n \in \{0,1\}$ within a vector that we define as the binary blendshape $\mathbf{b}_k = \{b_k^1, \ldots, b_k^N\}$. Based on our experiments in Section 5.2, we recommend a relative threshold of $25-40\%$ of the maximum displacement within the blendshape to decide if a vertex is active or non-active. We preferred a relative metric over an absolute one, as this scales better across subtle and strong expressions.

4.2. Optimization Problem

Creating a minimal set of reference expressions $\mathbf{x}_j$ for a blendshape transfer method can be defined as variation of a weighted set packing algorithm [Kar72]. Given a finite set that defines all unique elements (dictionary) together with a collection of subsets, each consisting only of a fraction of the unique elements, the set packing algorithm identifies a group of subsets that are pairwise disjoint (no unique element appears twice) and that have the maximum number of unique elements.

Applied to our problem, the list of vertex indices $\{1, \ldots, N\}$ is the finite set (dictionary), with every vertex index n defined as a unique element. Each binary blendshape $\mathbf{b}_k$ defines a subset of active vertices (vertices with strong displacements). The optimization problem (eq. 3) is to pack as many blendshapes as possible in one expression (Goal 1, Section 1), which is equivalent to maximizing the number of blendshapes with $w_k = 1$ (eq. 3a). At the same time, combinations of blendshapes should be prevented if their individual active vertices overlap (Goal 2). This can be expressed as a linear constraint (eq. 3b) for every vertex n. In the end only two states are relevant for a blendshape, either it is part of the expression or not. We observe in practice that it is easier to model and pose extreme expressions, rather than accurately extrapolating from in-betweens. While a closed-eye expression is well defined, small inaccuracies in a half-closed eye might negatively affect the closed-eye expression. For this reason w_k is always 0 or 1 (eq. 3c), making the problem an integer optimization problem. If only a limited set of example expressions can be provided, we would like to control which blendshapes should be chosen first (Goal 3). For this reason, we introduce the importance weight λ_k in eq. 3a. Details on computing λ_k are described in Section 4.3 and evaluated in Section 5.

$$\max_{w_k} \sum_{k=1}^{K} \lambda_k w_k, \tag{3a}$$

$$\text{s.t.} \sum_{k=1}^{K} b_k^n w_k \leq 1, \qquad \text{for all } n \in N \tag{3b}$$

$$w_k \in \{0,1\}, \qquad \text{for all } k \in K \tag{3c}$$

$$w^{\llcorner} - w^{\ulcorner} = 0, \quad \text{for all symmetry pairs } (w^{\llcorner}, w^{\ulcorner}) \tag{3d}$$

Algorithm 1 Expression Packing

In: J_{max} ▷ max number of reference expressions
$\mathcal{K} = \{1, \ldots, K\}$ ▷ set of available blendshapes
$\mathcal{R} = \{\}$ ▷ blendshapes in reference expressions
b = get_binary_blendshapes(μ) ▷ Section 4.1
λ_{Dis} = get_disp_strength() ▷ Section 4.3.1
λ_{TD} = get_tri_distortion() ▷ Section 4.3.2
find_symmetry_pairs() ▷ Section 4.4
while $\mathcal{K} \neq \emptyset$ **or** $J < J_{max}$ **do**
  λ_U = get_uniqueness() ▷ Section 4.3.3 (optional)
  $\lambda_k = \lambda_{Dis_k} + \lambda_{TD_k} + \lambda_{U_k}$
  $\mathcal{R} = \mathcal{R} \cup$ get_optimal_expression() ▷ Section 4.2
  $\mathcal{K} = \mathcal{K} \cap \mathcal{R}$

To increase the plausibility of the automatically computed reference expressions (Goal 4), we include symmetry constraints (eq. 3d). We observe from facial performance acting that it is easier to pose symmetric facial expressions (e.g., both eyes closed, both eyebrows frowning) than asymmetric facial expressions (e.g., left mouth smiling, right mouth sad). We enforce simultaneous activation of symmetric blendshapes, where $(w^{\llcorner}, w^{\ulcorner})$ are the blendshape weights of two symmetric blendshapes $(v^{\llcorner}, v^{\ulcorner})$. Section 4.4 provides more details on defining symmetric blendshape pairs.

Symmetry pairs and most importance weights are pre-computed once at the beginning. In contrast, eq. 3 is solved for every single reference expression (Algorithm 1) and returns the blendshape weights w_k^j for computing the reference expression $\mathbf{x}_j$. All blendshapes with $w_k = 1$ are removed from the set of available blendshapes $\mathcal{K}$ and the total number of blendshapes K is updated. The process is repeated until either a user-defined maximum number of expressions is reached or no blendshapes remain. The order of the computed reference expressions reflects their power to improve the overall result (Goal 3).

4.3. Importance Weighting of Blendshapes

The binary blendshape provides information about the activated vertices, but all blendshapes are considered as equally important if $\lambda_k = 1$. However, personalizing certain blendshapes is more important than others, which we model by computing an importance weight λ_k for every blendshape k in Equation 3. Let us analyze the origin of errors using example-based blendshape transfer techniques. First, semantically equivalent expressions can have individual variations, e.g., strong dynamic wrinkles in faces of old people versus barely visible wrinkles in young faces. While it is difficult to forecast exactly the individual differences of expressions, we can assume that subtle expressions remain subtle and expressions with strong movements have a higher chance to introduce noticeable individual differences due to strong deformations. We model this observation numerically by the displacement strength λ_{Dis}.

Second, for blendshapes that are missing training examples, deformation transfer [SP04] might introduce errors. Deformation transfer is based on the assumption that the shape of the template rig $\mathcal{A}$ is similar to the personalized rig $\mathcal{B}$ ($\mathbf{v}_0^A \approx \mathbf{v}_0^B$). As soon this assumption is violated, artifacts start to appear. Closer examination reveals that if triangles are non-uniformly scaled, results become inaccurate. Unfortunately, deformation gradients are only scale invariant in the tangential plane of a triangle [KG08] but not in the normal direction. Consequently, transferring the deformation between two shapes that are non-uniformly scaled, or where facial parts differ in their size relative to the head (e.g., big eyes vs. small eyes), will lead to visible artifacts as visualized in Figure 3. Notice as well that the non-similarity of meshes from the algorithm's perspective is different to the perceptual difference of characters. Simple non-uniform scaling creates highly distinctive characters for the deformation transfer algorithm, but not for a human. At the same time modifying the amount of fat in a human face leads to visually distinct faces, but not numerically, because facial parts most affected by blendshapes (e.g., eyes, mouth) remain the same. We aim to prioritize blendshapes which would be problematic to transfer by calculating the triangle distortion λ_{TD} between two characters.

Finally, we suggest an optional weighting term for blendshape uniqueness to to enforce distinctiveness of consecutive reference expressions. Although a large variation of expressions may appear advantageous in the first place, we observe that example based facial rigging [LWP10] tends to remove subtle blendshape differences, which we deemed important.

We observe that high-end character models with high numbers of blendshapes intentionally have subtle differences in their expressions (e.g., several versions of smiles), which is important to achieve high levels of realism. Blendshape transfer should maintain these subtle differences for high quality results. If, for example, we had 2 similar blendshapes in the model (e.g., evil and happy smile) with only one training example (e.g., happy smile) we lose nuances between the two smiles after blendshape transfer using the original alternating optimization (expression fitting and weight estimation) [LWP10]. Due to regularization, the weight estimation changes correct blendshape weights 0.0,1.0 to an incorrect e.g. 0.3,0.7. Enforcing correct blendshape weights 0.0,1.0 creates a generic evil smile, but more importantly an accurate fit of the happy

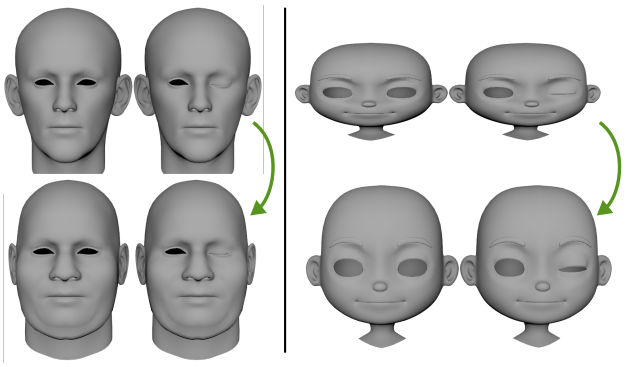

Figure 3: *Blendshape transfer using deformation transfer [SP04] for two character pairs. Left: Despite strong visual differences, results are accurate because of the non-changing shape of the eyes. Right: A simple non-uniform scale is a difficult case for expression transfer creating inaccurate results.*

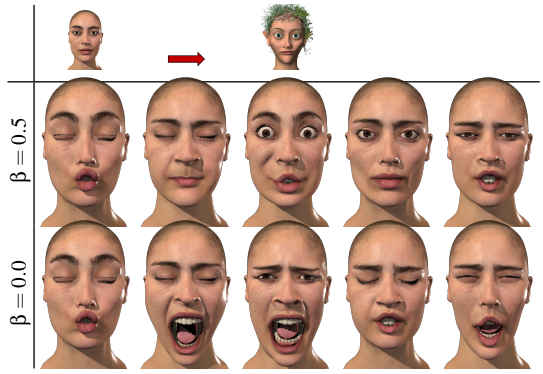

Figure 4: *First five expressions computed with ($\beta = 0.5$) and without ($\beta = 0.0$) the optional blendshape uniqueness metric.*

smile. Since neither situation is sufficient for the transfer of a high-end model, we consider the uniqueness term as optional (Figure 4). Each of these factors will be normalized and combined into a single factor, subject to user-defined variables (α, β). For our purposes we define the weights as $\alpha = 0.5, \beta = 0.0$ unless otherwise defined.

$$\lambda_k(\alpha,\beta) = \alpha\lambda_{Dis_k} + (1-\alpha)\lambda_{TD_k} + \beta\lambda_{Dist_k} \tag{4}$$

4.3.1. Displacement Strength

We consider the displacement weight of each blendshape, defined as the mean displacement of all vertices n between the blendshape $\mathbf{v}_k$ and the neutral expression $\mathbf{v}_0$. We consider only active vertices, as defined for binary blendshape $\mathbf{b}_k$ in Section 4.1. Notice that the dot product of $\mathbf{b}_k \cdot \mathbf{b}_k$ returns the number of active vertices. All mean displacements $\bar{d}_k$ are normalized to guarantee that λ_{Dis_k} is of range $[0,1]$.

$$\begin{aligned} \bar{d}_k &= \frac{1}{\mathbf{b}_k \cdot \mathbf{b}_k} \sum_{n=1}^{N} b_k^n \left\| \mathbf{v}_k^n - \mathbf{v}_0^n \right\| \\ \lambda_{Dis_k} &= \frac{\bar{d}_k}{\max\{\bar{d}_1, \ldots, \bar{d}_K\}} \end{aligned} \tag{5}$$

4.3.2. Triangle Distortion

We consider the triangle distortion between the neutral expressions of the template rig $\mathcal{A}$ and target character $\mathcal{B}$, since the relationship between the triangles of these two meshes is key for accuracy of deformation transfer. Given the equal connectivity between two blendshape models, we first compute the vectors $\mathbf{vu}$ and $\mathbf{vw}$, for each pair of triangles $\mathbf{uvw}$ of the meshes $\mathcal{A}$ and $\mathcal{B}$. The final distortion metric s between triangles t of the meshes $\mathcal{A}$ and $\mathcal{B}$ is then:

$$s^t = \max\left\{\frac{\mathbf{vu}_A}{\mathbf{vu}_B}, \frac{\mathbf{vu}_B}{\mathbf{vu}_A}\right\} + \max\left\{\frac{\mathbf{vw}_A}{\mathbf{vw}_B}, \frac{\mathbf{vw}_B}{\mathbf{vw}_A}\right\} \tag{6}$$

The variable s for triangles t is of range $[1,\infty]$ and is normalized in eq. 7 to ensure that λ_{TD} and λ_{Dis} are both of range $[0,1]$. Triangle distortion is only relevant for triangles affected by deformation within a blendshape. We multiply s^t, which is only based on the two neutral shapes, with the blendshape dependent b_k^t, which is 1 if at least one vertex of the triangle t is active and 0 otherwise.

$$\begin{aligned} s_k &= \sum_{t=1}^{T} b_k^t s^t, \quad b_k^t = b_k^u \cup b_k^v \cup b_k^w \\ \lambda_{TD_k} &= \frac{s_k}{\max\{s_1, \ldots, s_K\}} \end{aligned} \tag{7}$$

4.3.3. Blendshape Uniqueness

The Pearson coefficient quantifies similarity between two blendshapes [LAR*14, RZL*17]. The codomain of the Pearson coefficient is in the range $[-1,1]$, with 1 indicating two blendshapes are the same, 0 indicating they are different, and -1 indicating they are the same but in opposing directions (e.g. opening and closing of the mouth). As we wish to consider opposite movements as different, we change all negative coefficients to 0. The first reference expression is unique by definition such that $\lambda_{U_k} = 0$. All blendshapes that are part of a reference expression are saved in the set $\mathcal{R}$, with r being the index number and R the total number of blendshapes in $\mathcal{R}$. All other blendshapes remain in the set $\mathcal{K}$ (see Algorithm 1), with K being now the available blendshapes for packing and not the overall number of blendshapes anymore. To compute the uniqueness weight, we first sum the Pearson coefficient between a blendshape k and all blendshapes in $\mathcal{R}$. The result is normalized and inverted. The intuition behind this metric is, if no similar blendshape is part of the set of reference expressions the uniqueness weight is high and low otherwise.

$$\begin{aligned} U_k &= \frac{1}{R} \sum_{r=1}^{R} \max\left\{ \frac{\delta\mathbf{v_k} \cdot \delta\mathbf{v_r}}{\|\delta\mathbf{v_k}\| \, \|\delta\mathbf{v_r}\|}, 0 \right\}, \text{ if } R > 0 \\ \lambda_{U_k} &= 1 - \frac{U_k}{\max\{U_1, \ldots, U_K\}} \end{aligned} \tag{8}$$

4.4. Symmetry Constraints

Our overall goal is to propose reference expressions that are meaningful and have good numerical properties. To our knowledge, no method exists that can determine whether an expression can be posed by a human. Interestingly, even well trained actors can have difficulty in posing all defined Action Units individually [CKH11]. We discarded the idea of learning plausible expressions from animation data, because this would require the creation of an expressive animation sequence for every template rig, a time-consuming and difficult task. Furthermore, plausible expressions that are not part of the animation would be considered as infeasible, unnecessarily limiting the combination space of blendshapes. We preferred a data-free solution that largely ensures plausible expressions in combination with an optional artist friendly refinement feature (Section 4.5.3). This enables the user to more quickly accomplish the task of creating poseable expressions. Even with manual refinement, the workflow remains largely automated.

We observe that symmetric facial expressions are easier to perform and appear more frequently, e.g., closing eyes or lifting eyebrows simultaneously, a smile on both sides. We want to prefer such combinations because they strongly improve the likelihood to

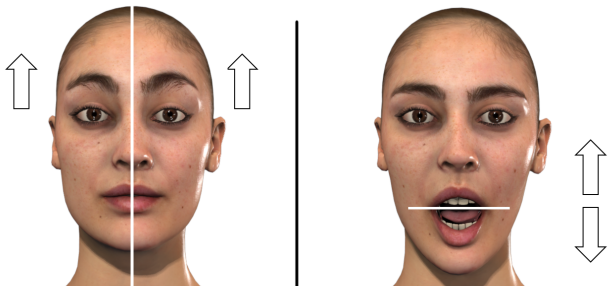

Figure 5: *Symmetric blendshape activation. Shape symmetry remains while lifting both eyebrows (left) and motion symmetry is created due to perpendicular movement of the lips (right).*

create visually plausible expressions (Goal 4). We identify two different types of facial symmetry: shape symmetry and motion symmetry (Figure 5). Expressions like closing both eyes, a smile etc., create shapes that are symmetric between the left and right side of the face. Apart from the left and right side, facial shapes are not symmetric. Nevertheless, the upper and lower lip move in coordination and certain symmetry exists between the main motion directions. The same applies for the lower eyelid and the upper eyelid together with the eyebrows. In the following, we aim to automatically identify blendshape pairs that contain symmetric activation of the face (e.g., a left eyebrow raise blendshape and a right eyebrow raise blendshape).

4.4.1. Shape Symmetry

For shape symmetry, we require a one-to-one mapping between the vertices on the left and right-hand side of the face. This mapping can be computed automatically, either based on the mesh topology [Aut16], where a user defines a single triangle edge as the symmetry axis, or independent of the mesh topology [MGP07] by comparing different samples of the mesh surface. For all our tested blendshape models, using symmetry detection based on topology was sufficient. Once the symmetry mapping is computed on the neutral mesh, our algorithm first automatically detects all self-symmetric blendshapes, (e.g., a smile that affects both sides of the face) and excludes them from the symmetry search. Intuitively, one might also consider excluding all blendshapes that have active vertices on both sides of the symmetry axis. However, this exclusion is too broad. Many blendshapes for the mouth and eyebrows have active vertices on both sides of the symmetry axis in combination with a symmetric opponent (see Figure 2).

For the remaining non-self-symmetric blendshapes, we project the displacements of blendshape m on the symmetry plane using the projection function $p(\delta\mathbf{v}_m^n)$ and compute the final difference between the potentially symmetric blendshape k and and the projected blendshape $p(m)$ using the following equation:

$$\varepsilon_{km} = \frac{1}{\sum_{n=1}^{N}\left(\mathbf{b}_k^n \cap p\left(\mathbf{b}_m^n\right)\right)} \sum_{n=1}^{N} \left\| \mathbf{b}_k^n \delta\mathbf{v}_k^n - \mathbf{b}_m^n\, p\left(\delta\mathbf{v}_m^n\right) \right\| \tag{9}$$

Notice that the intersection of $\mathbf{b}_k^n$ and $p(\mathbf{b}_m^n)$ is smaller than the union. While normalizing by the union would compute the average vertex displacement between $\mathbf{b}_k^n$ and $p(\mathbf{b}_m^n)$, normalizing by the intersection amplifies ε_{km} for non-symmetric blendshapes. If ε_{km} is below the following relative threshold, the blendshapes k and m are considered as symmetric.

$$\varepsilon_{km} < 0.4 \min_{n \in N} \left\{ \max \left\| \delta\mathbf{v}_k^n \right\|, \max \left\| \delta\mathbf{v}_m^n \right\| \right\} \tag{10}$$

When multiple symmetric blendshapes are found, we take the pair with the lowest error and check for possible overlapping of active vertices. We allow small overlap between symmetric pairs, as we have found that pairs of blendshapes that should be considered symmetric often have an overlap at the axis of symmetry (e.g., Figure 2, left). Blendshapes that spread along both facial halves will be not combined by the symmetry constraint. As our optimization algorithm strictly prevents any overlap between blendshapes in an expression, active vertices must be corrected for symmetric pairs of blendshapes in advance. Symmetric blendshapes will be forced to be included together, therefore the total active area remains the same.

4.4.2. Motion Symmetry

Finding a reliable mapping between vertices of the upper mouth/nose area and the lower mouth/chin area is difficult because facial parts differ significantly in terms of shape for the neutral pose and in terms of displacement for the blendshapes. However, defining a symmetry in these areas is important in order to prevent impossible combinations such as a kiss for the upper lips and a lip bite for the lower lips.

Lacking a promising automatic method, we use a sketch-based method, where the user selects the vertices of the two pairs of symmetric areas. The selection procedure is shown in the supplemental movie. As an approximate selection is sufficient, we rely on Maya's built-in mesh painting interface in our implementation. This allows the entire task to be accomplished within a minute. Blendshapes that either have no selected vertices or selected vertices from more than one sketch, are removed from the set of possible candidates for motion symmetry. The remaining blendshapes have very local deformations, a small number of active vertices and a dominant displacement direction. Lacking a reliable one-to-one mapping approach between individual vertices, we cannot compare per-vertex displacements as we did for shape symmetry. Instead, for every blendshape we compute an average displacement direction $\delta\bar{\mathbf{v}}_k$.

$$\delta\bar{\mathbf{v}}_k = \frac{1}{\left\| \sum_{n=1}^{N} \mathbf{b}_k^n \delta\mathbf{v}_k^n \right\|} \sum_{n=1}^{N} \mathbf{b}_k^n \delta\mathbf{v}_k^n \tag{11}$$

We then compare the average blendshape displacements between the potential symmetry pair candidates k and m,

$$\cos(\phi_{km}) = \delta\bar{\mathbf{v}}_k \cdot \delta\bar{\mathbf{v}}_m\,. \tag{12}$$

We consider all blendshapes with $\cos(\phi_{km})$ below a threshold of -0.985 as symmetric. The minimal threshold is equivalent to 10° difference in the opposite direction (asymmetric activation) and symmetry pairs are built based on the smallest angle. As previously, active vertices are modified to prevent possible overlapping.

Test-case	N	K	Optimal	Greedy
Male 	7366	73	4.97s	0.29s
Toon 	11680	65	7.14s	0.39s
Female 	5034	265	13.14s	0.39s
DigiDouble 	5131	161	13.14s	0.39s

Table 2: *Timings for computing the first reference expression. Computation time depends on the vertex number N and the number of blendshapes K in the blendshape rig.*

Test-case	5 ex.	10 ex.	20 ex.
Male	36/38 (31)	49/49 (42)	63/63 (56)
Toon	34/33 (22)	48/49 (43)	62/62 (61)
Female	58/54 (34)	89/85 (57)	124/116 (93)
DigiDouble	61/54 (39)	90/79 (55)	126/121 (104)

Table 3: *Number of blendshapes selected by greedy and optimal algorithms and their overlap in the first 5, 10 and 20 training expressions. Format: Greedy/Optimal (Overlap).*

4.5. Solving the Optimization Problem

Solving a set packing problem (eq. 3) is np-hard. However, by formulating the problem as an integer linear problem and using existing numerical libraries, the optimal solution can be computed within a reasonable time (Table 2). Alternatively, greedy approximation methods facilitate interactive framerates ($<$ 1s) and sufficient results. In the following, we discuss the details together with the advantages and disadvantages.

4.5.1. Optimal Solution

Our optimization problem (eq. 3) is a derivation of the weighted set packing problem, which is a classic example of an np-complete problem [Kar72]. We formulate it as a boolean linear programming problem, which is a special case of mixed integer linear programming (MILP) due to the linear objective function together with linear (in-)equality constraints and $w_k \in \{0,1\}$. We compute the optimal solution in Python using the PuLP library in combination with the numerical library CPLEX. The linear objective function consists of K unknowns, one for each blendshape. The number of linear constraints is $N+P$ which is the number of vertices and symmetry pairs $(w^{\llcorner}, w^{\ulcorner})$. Various timings for solving the linear programming problem are listed in Table 2.

The advantage of the optimal solution is that it computes the global optimum, e.g. by using a branch and cut algorithm. Computation time depends on the number of vertices, the number of blendshapes and the distribution of activated vertices across different blendshapes. For example, if blendshapes either activate all vertices in the upper or lower face halves, the number of combinations to test is less, than if different blendshapes activate vertices at different locations. This blendshape-specific component makes it difficult to predict timings exactly for different blendshape rigs. Overall, we notice that the optimal solution is slower compared to the greedy solution.

4.5.2. Greedy Approximation

We implement Kordalewski's greedy set packing algorithm [Kor13] with some alterations to suit our method as shown in Algorithm 2. First, we select blendshapes based on the smallest weighted sum of active vertices, where the importance weight λ_n is inverted. If the weight is constant across all blendshapes and vertices, this fits the largest number of blendshapes into one expression. Second, we include an option to allow a small percentage of overlap between blendshapes, which is not possible in case of the optimal solution.

Algorithm 2 Greedy Set Packing

```
x_j = v_0                                   ▷ example expression
K = {1,...,K}                               ▷ non-covered blendshape indices
λ_max = 1                                   ▷ maximum possible weight
for k in K do
    n_k = b_k · b_k                         ▷ sum active vertices
    λ_k n_k = (λ_max − λ_k) n_k             ▷ precompute importance
while K ≠ ∅ do                              ▷ expression packing
    δv^L = smallest( λ_k n_k)
    δv^⌜ = find_symmetric(δv^L)
    x_j += δv^L + δv^⌜                      ▷ add blendshapes
    K = K \ {k^L, k^⌜}                      ▷ remove covered blendshapes
    for m in K do                           ▷ remove overlapping blendshapes
        if overlap(x_j, b_m) then
            K = K \ {m}
return x_j
```

4.5.3. Artist Friendly Refinement

After incorporating the symmetry constraint and fine-tuning the importance weights, the suggested expressions by our algorithm are plausible in about 85% of cases (see Section 5). To correct remaining implausible expressions but still maintain the optimal selection of blendshapes, we allow for artist intervention in our greedy algorithm. As each blendshape is chosen, the user is presented with the options to accept or reject it. If blendshapes are rejected, the algorithm proposes alternative nearly optimal blendshapes and the process is repeated. All rejected blendshapes are no longer considered for the current expression, but will be reconsidered when computing the next expression.

5. Results and Evaluation

In this section, we evaluate the influence of different parameters and solvers on the final output and the pose-ablity of the expressions. For evaluation, we use a derivation of the original example-based blendshape transfer method [LWP10]. The method required a number of training expressions in combination with estimates of the blendshape weights. By design, the blendshape weights are 0 or 1 and are known for our training expressions. We therefore remove the weight estimation step within the alternating optimization (fitting towards examples vs. blendshape weight estimation).

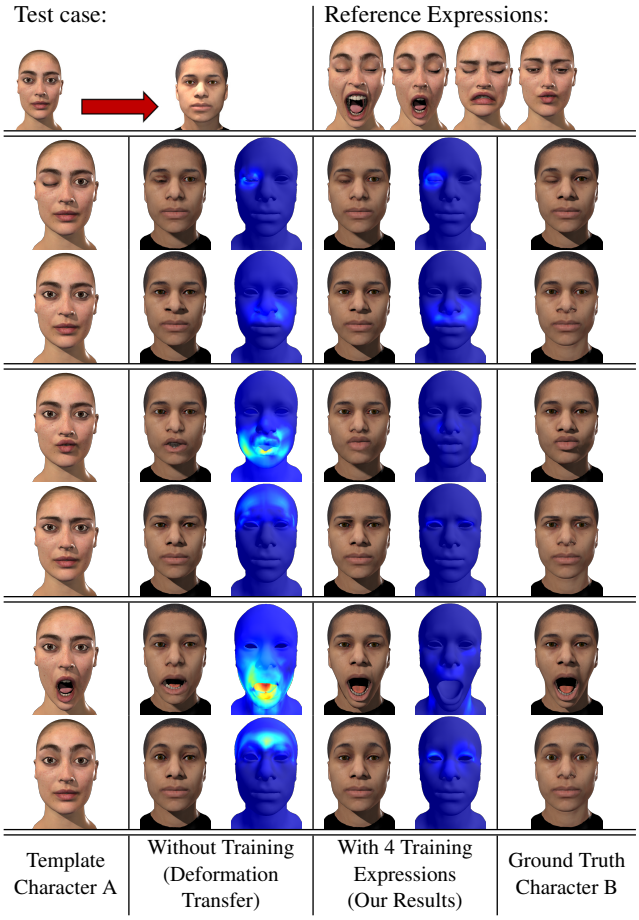

Figure 6: *Comparison of individual blendshapes after applying blendshape transfer without training expressions (Deformation Transfer) and with only four training expressions based on our reference expressions. Each row shows two blendshapes that have been corrected by one training expression.*

5.1. Dataset

One immanent challenge of evaluating example-based blendshape transfer methods are suitable character assets. For ground truth comparisons, two blendshape models $\mathcal{A}$ and $\mathcal{B}$ are required of equal vertex connectivity and semantically equivalent blendshapes. In practice, either the total number of blendshapes is small [CWZ*14] or a small number of blendshapes is identical between two characters, while other blendshapes can range from similar to completely different. This is even the case in recent datasets, e.g., 3D FACS and FLAME [CKH11, LBB*17]. Furthermore, evaluation should cover difficult cases, meaning that the two blendshape models should have significantly different proportions in areas that are deformed most by blendshapes (Section 3).

As a ground-truth test for the digital double use-case, we acquired two high-end photogrammetry-scanned characters, created by Eisko, a leading Digital Double company (Figure 6). We refer to these characters as Female 1 and Female 2. Semantically equivalent blendshapes between both models were selected manually. Non-equivalent blendshapes were removed. This resulted in a total of 161 in-correspondence blendshapes between Female 1 and Female 2. To establish equal connectivity between the two high-quality facial rigs we followed the default procedure as described previously (Figure 1). We refer to this test-case as DigiDouble. Because the two rigs are based on real people and 3d scans, this is the closest possible approximation to a real use-case that offers ground truth comparisons. In such a dataset, the blendshape weights of the reference expressions can be copy-pasted to obtain training expressions. After running the example-based blendshape transfer algorithm with only four training expressions, we compare the individual blendshapes for character $\mathcal{B}$ with the original (see Figure 6, right columns). Despite a compact packing of over 30 blendshapes in a few expressions visible artifacts that appeared when using deformation transfer only, are removed and only small numerical errors appear where blendshapes overlapped. A larger set of results can be found in the appendix in Figure 14.

To provide more test-cases, especially of very challenging scenarios, we first gathered 3 production-quality rigs: a Male, Female (Female 1 from before), and a Toon (see Table 2 and Figure 11). The characters ranged in vertex count (5034 to 11,680), number of blendshapes (from 65 to 265), and whether they were sculpted by hand (Toon and Male) or high-end photogrammetry-scanned (Female). Since it is difficult to obtain rigs with semantically identical blendshapes, we generated altered versions of our rigs, by applying a set of deformations to the entire blendshape model. This includes non-uniform scaling as well as local enlarging and shrinking of facial parts like eyes and mouth using free-form deformation. These alterations were specifically designed to generate proportionally non-similar character pairs, which are difficult cases for deformation transfer (see Figure 3, right). To simplify naming, test-cases using the original character and warped version are named as the original character (Toon, Male, Female).

5.2. Generic Optimization Parameters

Our proposed optimization has a set of user-parameters to refine the computation of reference expressions. To avoid unnecessary parameter tweaking, we investigate suitable default parameters that create good results for all test cases. In general, we aimed in previous sections for relative metrics in order to be scale independent and added normalization to be more stable across character rigs with different numbers of vertices or blendshapes.

Binary Blendshapes Threshold A relative threshold (eq. 2) turned out to be the best compromise for defining active vertices for both subtle and expressive blendshapes. For evaluation, we tested different values of μ on characters with few and many blendshapes. During the evaluation we set $\lambda_k = 1.0$ in eq. 3.Increasing μ increases the number of blendshapes within one reference expression and decreases the total number of expressions. At the same time, a high value for $\mu > 0.4$ created frequently unfeasible expressions for blendshape models with $K \approx 50$ and could drop to $\mu > 0.3$ for blendshape models with $K \approx 265$ (Figure 7). During parameter testing of the maximum overlap between two blendshapes, we also noticed that a higher overlap creates a bigger error during the reconstruction of individual blendshapes from training expressions.

Figure 7: *Output from expression packing algorithm for our Female test-case: the first four computed reference expressions using different thresholds for the definition of binary blendshapes. A threshold of up to $\mu = 0.3$ creates plausible results in most cases even for model with many blendshapes ($K = 265$).*

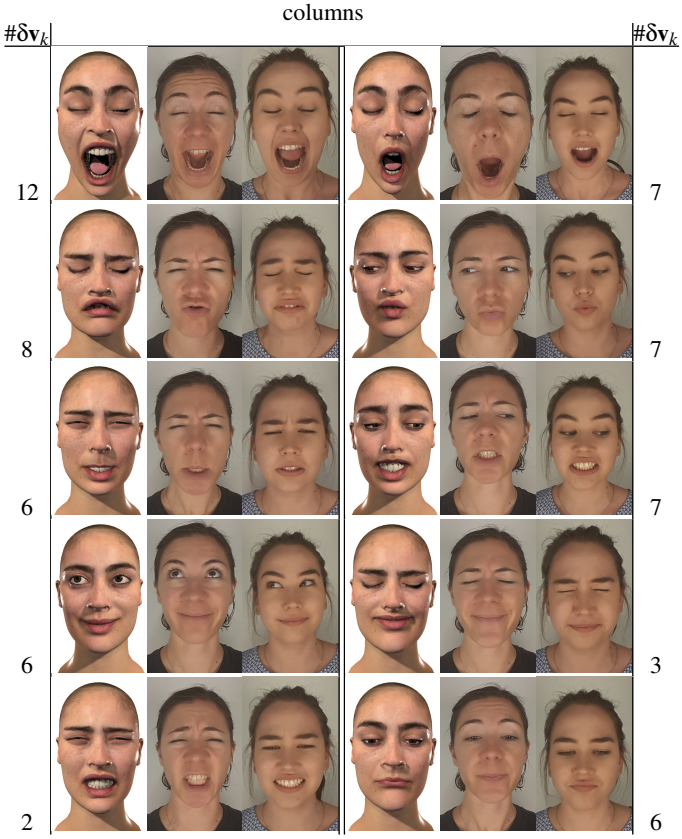

Figure 8: *Two volunteers demonstrating the pose-ability of the DigiDouble expressions. The number of blendshapes in each expression is shown on the left and right columns.*

We therefore recommend $0.25 \leq \mu \leq 0.4$ as a good trade-off between the smallest number of examples and plausible examples. In the remaining evaluation, we use $\mu = 0.3$. An example of a combined expression with its constituent blendshapes can be found in the Appendix in Figure 13.

Importance Weighting For evaluation of the importance weights, we linearly interpolate between different importance weights: $\lambda = \alpha\lambda_{Dis} + (1.0 - \alpha)\lambda_{TD}$. Symmetry constraints are removed during numerical comparisons to avoid any bias. Remember that a small nmber of vertices were allowed to overlap for symmetric blendshapes. As a measure of improvement, we compute the root mean squared distance between the ground truth blendshapes of the target model and the output of an example-based blendshape transfer algorithm (with training expressions as input). As intended, adding importance weights decreases the error (Figure 9). Weighting both importance values equally ($\alpha = 0.5$) was found to be the best trade-off across different characters and numbers of training expressions. This confirms that both importance metrics are relevant.

Greedy vs Optimal Finally, we compare the reference expressions computed by the optimal and the greedy solver (Section 4.5). Between 67% and 98% of blendshapes selected by the optimal method have also been selected by the greedy solver (Table 3), meaning that both solvers compute similar, but not identical results. The greedy algorithm creates reference expressions with slightly more blendshapes due to the permission of little overlap between active vertices and is nearly 20 times faster (Table 2). In contrast, blendshape transfer in combination with training expressions of the optimal solver creates blendshapes that are closer to ground-truth in terms of RMS-error (See Figure 10). With only 20 training expressions, both methods covered approximately 67% blendshapes for Female, 86% for Male and 95% for Toon, which is a remarkable reduction of training expressions. Interestingly, adding the symmetry constraint to the optimal solution had basically no influence on the measured RMS-error, while the RMS-error increased for the greedy algorithm with active symmetry constraints.

5.3. Expression Pose-ability

Good reference expressions should not only show good numerical properties, but should be poseable by an actor, to facilitate scanning. The general unweighted set packing algorithm creates reference expressions that might be difficult or even impossible to pose (Figure 11, left). Our results showed that adding importance weighting had a positive side-effect on expression plausibility (Fig-

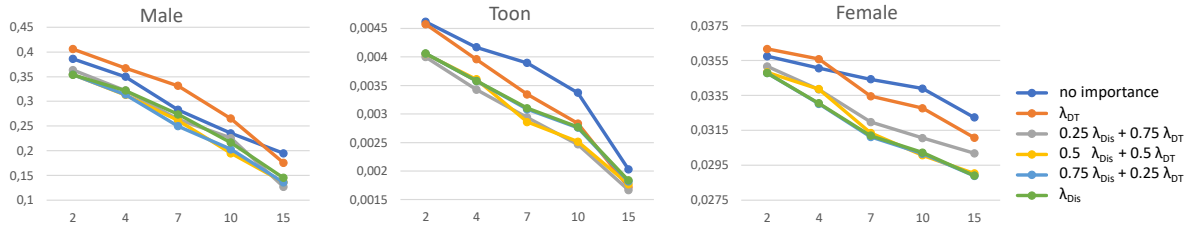

Figure 9: *Average RMS-error per vertex between ground truth and the output of example-based blendshape transfer for 3 different test-cases. Number of training expressions ranges from 2 to 15, (shown on x-axis). Importance weighting that considers both, vertex displacement (λ_{Dis}) and triangle distortions (λ_{TD}), creates most accurate results across different characters and number of training expressions.*

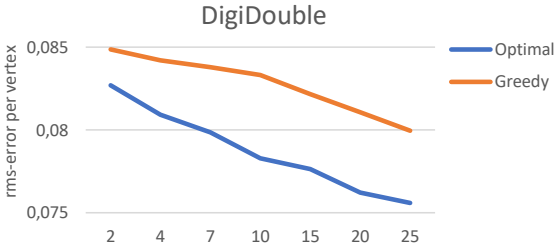

Figure 10: *RMS-error for our DigiDouble test case ($k = 161$) with different numbers of training expressions as input (ranging from 2 to 25, shown on x-axis). Training expressions were computed with the optimal and greedy algorithms.*

ure 11, middle). Blendshapes with strong displacements, which coincidentally tend to have more active vertices, started to be part of the first reference expressions. Rather than packing several subtle expression in the first reference expressions, subtle and expressive blendshapes were distributed more evenly. Some reference expressions even appeared symmetric, due to the fact that some template rigs were perfectly symmetric, and therefore left and right blendshapes had equal importance factors. If these blendshapes do not overlap, then they are included at the same time, thus creating already symmetric expressions.

To create more plausible expressions automatically, we introduced the symmetry constraint (Figure 11, right). To evaluate the final output of our Expression Packing algorithm, we recruited two volunteers to demonstrate posing of the first ten reference expressions of the DigiDouble case. The first expressions are the most tightly packed and therefore the most difficult to pose. Some expressions were slightly modified because the symmetry constraint at times forced the eyes to look in opposite directions. After editing, both eyes looked in the same direction and both volunteers were able to pose all ten expressions within short capturing sessions (Figure 8).

6. Discussion

The main novelty of our work is the formulation of the problem of finding optimal reference expressions as a special case of integer linear programming. Our expression packing algorithm works best with locally defined blendshapes. Blendshapes modelling entire emotions (e.g., happy, angry, visemes, etc.) or large parts of the face (e.g., entire eyes and eyebrows) are more difficult to combine with other expressions. The automatically created reference expressions are largely plausible even for challenging high-quality rigs consisting of over 250 blendshapes. Furthermore, our artist-friendly refinement method gives the user full control over the created reference expressions, without the need to track minimal overlap of blendshapes or estimate the influence on the optimization for a blendshape. It is also possible to further reduce the number of expressions within a feedback loop by at the cost of increasing the overlap between blendshapes. Overall, the reduction might be small because our packing is already very compact (Table 3) and a globally optimal solution is found.

Other practical and easy to integrate extensions of our method could be: First, limit the maximum number of blendshapes within a reference expression to achieve a more equal distribution of the total number of blendshapes per reference expression. This can be formulated as a linear constraint in eq. 3. Second, check in a second step whether any blendshapes that are already part of a reference expression could be added to the most recently computed reference expression. This would allow for adding additional examples of blendshapes without increasing the overall number of reference expressions.

In the future, we would like to apply our method to a number of characters with equivalent FACS-based rigs in order to propose a generalized expression set which would be applicable to all FACS-based rigs. However, a general set of reference expressions comes at the cost of not being optimal for each individual, because the triangle distortion metric between neutral expressions must be ignored. Also, combining perceptual and numerical metrics for importance weights is an interesting direction for future work.

In our test-cases, building upon Maya's shape symmetry functionality was sufficient to obtain symmetric pairs of blendshapes, even for non-symmetric character rigs that were built from scans. However, for very asymmetric faces (see Figure 12), that are uncommon as template models, our symmetry detection would not be accurate enough. It is possible that more advanced symmetry detection tools (e.g., [MGP07]) would improve the accuracy. The symmetry constraint greatly improved expression plausibility. Additional metrics could potentially improve the plausibility even further, such as physical constraints, prevention of self-intersection or anatomical constraints (e.g., eye-gaze direction).

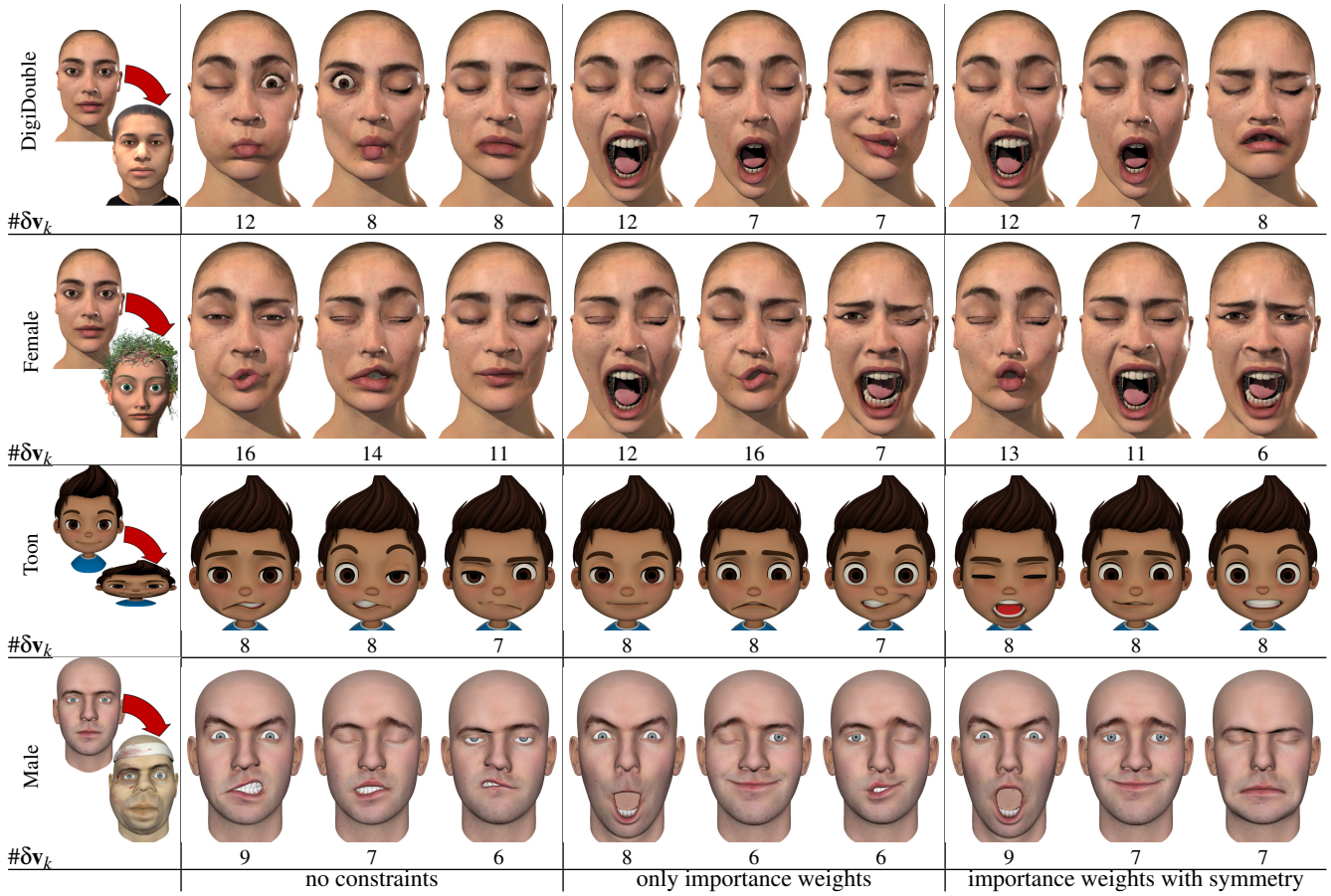

Figure 11: *Output from our expression packing algorithm showing the first 3 computed reference expressions. The number below each is the number of blendshapes that have been packed into that reference expression. Expressions were computed (left) with no importance weighting or symmetry constraints, (middle) with importance weights only, and (right) with importance– weights and symmetry constraints.*

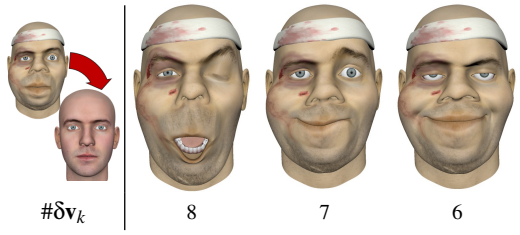

Figure 12: *Reference expressions computed for a highly asymmetric character (an unusual template model).*

7. Conclusion

In this paper, we proposed a solution for automatic reference expression creation, facilitating optimal training expressions for example-based blendshape transfer. This is achieved through constrained blendshape packing. Our key contributions include blendshape importance ordering specifically aiming to deal with the weaknesses of the deformation transfer algorithm, and the use of the weighted set packing algorithm for creation of information-dense reference expressions. Combined, these produce as-few-as-possible packed reference expressions, tailored to the characters and blendshapes to which the blendshape transfer will be applied. Using the recommended parameters, our algorithms achieves a remarkable density of blendshape information. With 20 examples, our results showed that almost the full set of blendshapes of our test rigs were covered and 67% of the blendshapes of our highly-expressive production rig.

Acknowledgements

This research was funded by Science Foundation Ireland under the Game Face project (Grant 13/CDA/2135) and the ADAPT Centre for Digital Content Technology (Grant 13/RC/2106).

References

[Aut16] AUTODESK: Maya, 2016. URL: www.autodesk.com/maya. 7

[BP14] BOUAZIZ S., PAULY M.: Semi-supervised facial animation retargeting, 2014. 3

[BV*99] BLANZ V., VETTER T., ET AL.: A morphable model for the synthesis of 3d faces. In *Siggraph* (1999), vol. 99, pp. 187–194. 2

[BVGP09] BARAN I., VLASIC D., GRINSPUN E., POPOVIĆ J.: Semantic deformation transfer. In *ACM Transactions on Graphics (TOG)* (2009), vol. 28, ACM, p. 36. 2

[BWP13] BOUAZIZ S., WANG Y., PAULY M.: Online modeling for realtime facial animation. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 40:1–40:10. 2

[CFA*16] CASAS D., FENG A., ALEXANDER O., FYFFE G., DEBEVEC P., ICHIKARI R., LI H., OLSZEWSKI K., SUMA E., SHAPIRO A.: Rapid photorealistic blendshape modeling from rgb-d sensors. *Computer Animation and Virtual Worlds* (2016). 3

[CHMA18] CARRIGAN E., HOYET L., MCDONNELL R., AVRIL Q.: A preliminary investigation into the impact of training for example-based facial blendshape creation. In *Eurographics Short Papers* (2018). 3

[CHZ14] CAO C., HOU Q., ZHOU K.: Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 43:1–43:10. 2

[CKH11] COSKER D., KRUMHUBER E., , HILTON A.: A facs valid 3d dynamic action unit database with applications to 3d dynamic morphable facial modeling. pp. 2296–2303. 6, 9

[CWZ*14] CAO C., WENG Y., ZHOU S., TONG Y., ZHOU K.: Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (Mar. 2014), 413–425. 3, 9

[DN08] DENG Z., NOH J.: Computer facial animation: A survey. In *Data-driven 3D facial animation*. Springer, 2008, pp. 1–28. 2

[EF78] EKMAN P., FRIESEN W. V.: *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists Press, 1978. 3

[FDH*15] FRIED O., DIVERDI S., HALBER M., SIZIKOVA E., FINKELSTEIN A.: Isomatch: Creating informative grid layouts. *Computer Graphics Forum 34*, 2 (2015), 155–166. 3

[FNH*17] FYFFE G., NAGANO K., HUYNH L., SAITO S., BUSCH J., JONES A., LI H., DEBEVEC P.: Multi-view stereo on consistent face topology. *Comput. Graph. Forum 36*, 2 (May 2017), 295–309. 1, 2, 3

[FSF07] FRATARCANGELI M., SCHAERF M., FORCHHEIMER R.: Facial motion cloning with radial basis functions in mpeg-4 fba. *Graphical Models 69*, 2 (2007), 106–118. 2, 3

[GYQ*18] GAO L., YANG J., QIAO Y.-L., LAI Y.-K., ROSIN P. L., XU W., XIA S.: Automatic unpaired shape deformation transfer. In *SIGGRAPH Asia 2018 Technical Papers* (2018), ACM, p. 237. 2

[GZC*16] GARRIDO P., ZOLLHÖFER M., CASAS D., VALGAERTS L., VARANASI K., PÉREZ P., THEOBALT C.: Reconstruction of personalized 3d face rigs from monocular video. *ACM Transactions on Graphics (TOG) 35*, 3 (May 2016), 28:1–28:15. 3

[HKG11] HUANG Q., KOLTUN V., GUIBAS L.: Joint shape segmentation with linear programming. *ACM Trans. Graph. 30*, 6 (Dec. 2011), 125:1–125:12. 3

[HSW*17] HU L., SAITO S., WEI L., NAGANO K., SEO J., FURSUND J., SADEGHI I., SUN C., CHEN Y.-C., LI H.: Avatar digitization from a single image for real-time rendering. *ACM Transactions on Graphics (TOG) 36*, 6 (Nov. 2017), 195:1–195:14. 3

[IBP15] ICHIM A. E., BOUAZIZ S., PAULY M.: Dynamic 3d avatar creation from hand-held video input. *ACM Transactions on Graphics (TOG) 34*, 4 (July 2015), 45:1–45:14. 1, 2

[IKNDP16] ICHIM A.-E., KAVAN L., NIMIER-DAVID M., PAULY M.: Building and animating user-specific volumetric face rigs. In *Proc. Symp. on Computer Animation* (2016), pp. 107–117. 2

[Kar72] KARP R. M.: Reducibility among combinatorial problems. *Complexity of Computer Computations* (1972), 85–103. 4, 8

[KG08] KIRCHER S., GARLAND M.: Free-form motion processing. *ACM Transactions on Graphics (TOG) 27*, 2 (May 2008), 12:1–12:13. 5

[Kor13] KORDALEWSKI D.: New greedy heuristics for set cover and set packing. *arXiv preprint arXiv:1305.3584* (2013). 8

[LAR*14] LEWIS J. P., ANJYO K., RHEE T., ZHANG M., PIGHIN F. H., DENG Z.: Practice and theory of blendshape facial models. *Eurographics (State of the Art Reports) 1*, 8 (2014), 2. 2, 6

[LBB*17] LI T., BOLKART T., BLACK M. J., LI H., ROMERO J.: Learning a model of facial shape and expression from 4d scans. *ACM Transactions on Graphics (TOG) 36*, 6 (Nov. 2017), 194:1–194:17. 1, 2, 3, 9

[LMX*08] LIU X., MAO T., XIA S., YU Y., WANG Z.: Facial animation by optimized blendshapes from motion capture data. *Computer Animation and Virtual Worlds 19*, 3-4 (2008), 235–245. 3

[LWP10] LI H., WEISE T., PAULY M.: Example-based facial rigging. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 32. 1, 2, 3, 4, 5, 8

[LYYB13] LI H., YU J., YE Y., BREGLER C.: Realtime facial animation with on-the-fly correctives. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 42:1–42:10. 2

[MGP07] MITRA N. J., GUIBAS L. J., PAULY M.: Symmetrization. *SIGGRAPH 2007* (2007). 7, 11

[NN01] NOH J., NEUMANN U.: Expression cloning. In *Proc. of SIGGRAPH* (2001), pp. 277–288. 2

[OBP*12] ORVALHO V., BASTOS P., PARKE F., OLIVEIRA B., ALVAREZ X.: A facial rigging survey. In *Eurographics State of the Art Reports* (2012). 2

[Osi07] OSIPA J.: *Stop Staring: Facial Modeling and Animation Done Right*, second ed. Wiley Publishing, 2007. 2

[OZS08] ORVALHO V., ZACUR E., SUSIN A.: Transferring the rig and animations from a character to different face models. *Computer Graphics Forum 27*, 8 (2008), 1997–2012. 2, 3

[Pan03] PANDZIC I. S.: Facial motion cloning. *Graphical Models 65*, 6 (2003), 385–404. 2, 3

[PCNS05] PARK B., CHUNG H., NISHITA T., SHIN S. Y.: A feature-based approach to facial expression cloning: Virtual humans and social agents. *Comput. Animat. Virtual Worlds 16*, 3-4 (July 2005), 291–303. 3

[PHL*06] PIGHIN F., HECKER J., LISCHINSKI D., SZELISKI R., SALESIN D. H.: Synthesizing realistic facial expressions from photographs. In *ACM SIGGRAPH 2006 Courses* (2006), ACM, p. 19. 2, 3

[PKC*03] PYUN H., KIM Y., CHAE W., KANG H. W., SHIN S. Y.: An example-based approach for facial expression cloning. In *Proc. of Symposium on Computer Animation* (2003), SCA, pp. 167–176. 3

[RZL*17] RIBERA R. B. I., ZELL E., LEWIS J. P., NOH J., BOTSCH M.: Facial retargeting with automatic range of motion alignment. *ACM Transactions on Graphics (TOG) 36*, 4 (July 2017), 154:1–154:12. 3, 6

[Sai13] SAITO J.: Smooth contact-aware facial blendshapes transfer. In *Proceedings of the Symposium on Digital Production* (2013), DigiPro '13, pp. 7–12. 2

[SCSN11] SONG J., CHOI B., SEOL Y., NOH J.: Characteristic facial retargeting. *Computer Animation and Virtual Worlds 22*, 2-3 (2011), 187–194. 3

[Sey16] SEYMOUR M.: Put your (digital) game face on, 2016. URL: https://www.fxguide.com/featured/put-your-digital-game-face-on/. 1

[Sey19] SEYMOUR M.: Weta digital's remarkable face pipeline : Alita battle angel, 2019. URL: https://www.fxguide.com/featured/weta-digitals-remarkable-face-pipeline-alita-battle-angel/. 1

[SILN11] SEO J., IRVING G., LEWIS J. P., NOH J.: Compression and direct manipulation of complex blendshape models. *ACM Transactions on Graphics (TOG) 30*, 6 (2011), 164:1–164:10. 3

[SL14] SEO J., LEWIS J. P.: Developing interactive facial rigs in production environment. In *ACM SIGGRAPH 2014 Talks* (2014), SIGGRAPH '14, pp. 36:1–36:1. 3

[SLS*12] SEOL Y., LEWIS J., SEO J., CHOI B., ANJYO K., NOH J.: Spacetime expression cloning for blendshapes. *ACM Transactions on Graphics (TOG) 31*, 2 (Apr. 2012), 14:1–14:12. 3

[SML16] SEOL Y., MA W.-C., LEWIS J. P.: Creating an actor-specific facial rig from performance capture. In *Proceedings of the 2016 Symposium on Digital Production* (2016), DigiPro '16, pp. 13–17. 1, 3, 4

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)* (2004), vol. 23, ACM, pp. 399–405. 1, 2, 5

[TZN*15] THIES J., ZOLLHÖFER M., NIESSNER M., VALGAERTS L., STAMMINGER M., THEOBALT C.: Real-time expression transfer for facial reenactment. *ACM Transactions on Graphics (TOG) 34*, 6 (2015). 2

[VLR*17] VESTNER M., LITMAN R., RODOLA E., BRONSTEIN A., CREMERS D.: Product manifold filter: Non-rigid shape correspondence via kernel density estimation in the product space. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2017). 3

[WBLP11] WEISE T., BOUAZIZ S., LI H., PAULY M.: Realtime performance-based facial animation. *ACM Transactions on Graphics 30*, 4 (2011), 77:1–77:10. 2

[WFLW18] WU W., FAN L., LIU L., WONKA P.: Miqp-based layout design for building interiors. *Computer Graphics Forum 37*, 2 (2018), 511–521. 3

[Won18] WONKA P.: Integer programming for layout problems. In *SIGGRAPH Asia 2018 Courses* (2018), SA '18, pp. 10:1–10:38. 3

[ZSCS08] ZHANG L., SNAVELY N., CURLESS B., SEITZ S. M.: Spacetime faces: High-resolution capture for˜ modeling and animation. In *Data-Driven 3D Facial Animation*. Springer, 2008, pp. 248–276. 3

Appendix

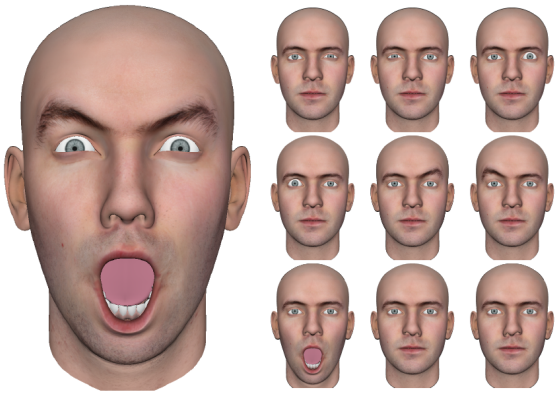

Figure 13: *First packed reference expression (Figure 11) together with the nine individual blendshapes.*

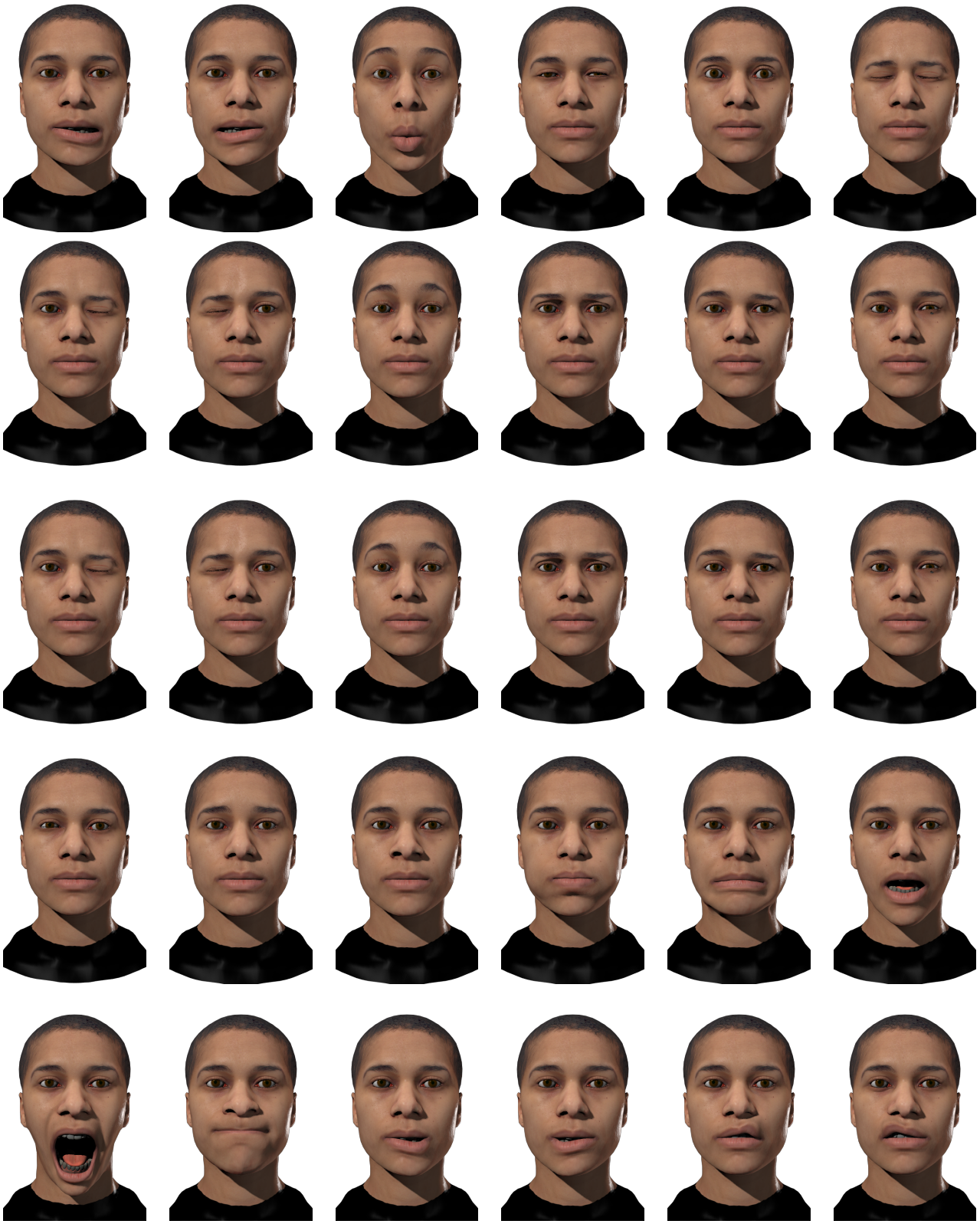

Figure 14: *Selection of individual blendshapes after applying blendshape transfer in combination with a full set of training expressions, where every blendshape is present in one of the training expressions.*