Landmark-guided deformation transfer of template facial expressions for automatic generation of avatar blendshapes

Hayato Onizuka Diego Thomas Hideaki Uchiyama Rin-ichiro Taniguchi
Kyushu University, Fukuoka, Japan
onizuka@limu.ait.kyushu-u.ac.jp, thomas@ait.kyushu-u.ac.jp
uchiyama@limu.ait.kyushu-u.ac.jp, rin@kyudai.jp

Abstract

Blendshape models are commonly used to track and re-target facial expressions to virtual avatars using RGB-D cameras and without using any facial marker. When using blendshape models, the target avatar model must possess a set of key-shapes that can be blended depending on the estimated facial expression. Creating realistic set of key-shapes is extremely difficult and requires time and professional expertise. As a consequence, blendshape-based re-targeting technology can only be used with a limited amount of pre-built avatar models, which is not attractive for the large public. In this paper, we propose an automatic method to easily generate realistic key-shapes of any avatar that map directly to the source blendshape model (the user is only required to select a few facial landmarks on the avatar mesh). By doing so, captured facial motion can be easily re-targeted to any avatar, even when the avatar has largely different shape and topology compared with the source template mesh. Our experimental results show the accuracy of our proposed method compared with the state-of-the-art method for mesh deformation transfer.

1. Introduction

Digital avatars are virtual representations of oneself. They allow people from all over the world to communicate in the digital world in a natural and human way. For example, avatars are extensively used in online games and Social Network Systems (SNS). While preserving the anonymity of users is a fundamental right of the Internet users, it is also desirable to have virtual avatars that are able to transmit true emotions. With true emotions, it is easier to communicate, and also to make the difference being humans and fake characters. Facial expressions are key features of the human emotions that do not tell anything about the person identity. Therefore they are the key to emotive virtual avatars.

Transferring facial expressions to an avatar mesh is called re-targeting. With the recent advances in facial feature detection [21, 22] and facial expression tracking [18, 12], several emotive avatars have been proposed by the main operators of social network systems (*e.g.*, Apple's animoji, Huawei's Qmoji, Google's Playmoji or Samsung's AR emoji). However, these emotive avatars are all pre-built and it is not yet possible to make one's favorite avatar become emotive. This is problematic because many applications and games actually base their business plan on selling and personalizing avatars (e.g., Fortine). Therefore, instead of changing all the avatars to a new (restricted) set of emotive avatars, it would be better to enable facial expression transfer to any personalized 3D avatar.

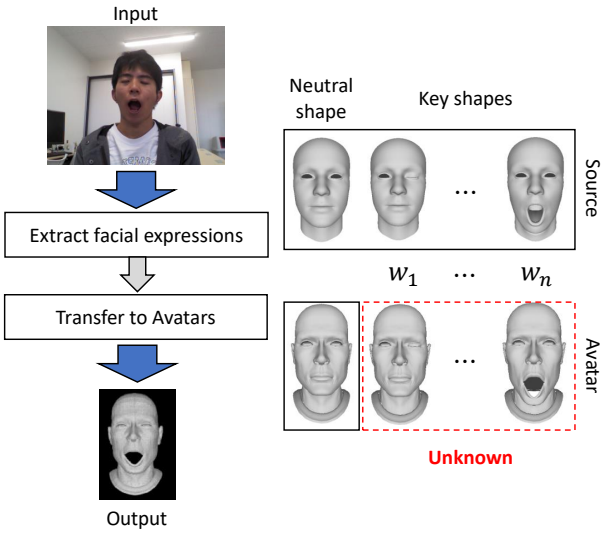

Figure 1: The general process of facial expression re-targeting using blendshapes.

There are several techniques that already exist to transfer facial expression to an avatar, which are frequently used in the movie and game industry. The most common technique is to use facial markers that can be easily tracked. Their motion can then be re-targeted to the corresponding

avatar landmarks [15]. Other approaches directly apply the transformation of the input expression to the avatar mesh [1, 3, 17]. These techniques allow to obtain precise and realistic re-targeting effects, but are restricted to professional usage. For the general user, it is too constraining and too difficult to correctly put a set of facial markers before entering into a SNS discussion, or the required calculation cost is too high for real-time re-targeting.

Marker-less re-targeting methods with light computational cost are needed for general usage. The state-of-the-art methods use blendshapes [10] to solve this problem. As shown in fig. 1, the blendshape strategy is to represent a facial expression by linearly blending several key expressions (also called key shapes). By estimating the weight of each key shape so that the source template model fits the user face, the virtual avatar can be animated by using the same weights given the corresponding key shapes.

Two strategies exist to map the tracked blendshape weights to the avatar blendshapes. The first strategy assumes that the set of key shapes are the same (in a semantic meaning). The second strategy maps the blendshape coefficients to the avatar key shape space with a mapping function that is either manually fitted or trained with a machine learning technique. For both strategies, the crucial point is that the avatar mesh must possess a set of key shapes that spans the space of all possible facial expressions. Creating such set of key shapes for any avatar mesh can be extremely difficult.

In general, it is the work of artists and animation experts to create blendshape models of a 3D mesh by using complex professional software. This process requires time and expertise, and it cannot be done by the general user. In this work we propose an automatic method to generate the key shapes of any facial 3D avatar. By simply selecting some facial landmarks in the avatar mesh, any user can instantly generate a set of key shapes that match the key shapes on the source template model used for tracking. Our proposed method has two main advantages: (1) it allows to create blendshape models easily, without any expertise or use of complex software; (2) because the key shapes matches the template model, the re-targeting process becomes easy and robust.

Our contributions are three-fold: (1) we propose a method for robust and accurate 3D mesh polygon correspondences; (2) we propose a new landmark-guided deformation transfer method; (3) we show application possibilities with re-targeting.

2. Related works

Creating facial animation from actors performances is a well studied problem with famous applications in the cinema and video game industry. In this section we focus on re-targeting and animation methods targeting the general consumer and that can be deployed on cheap devices.

Although early works have focused on creating facial animation using a combination of motion capture data and blendshape interpolation ([6, 7, 15]), recent techniques do not require facial markers anymore ([2, 5, 11]). To do this Dutreve *et.al.*, [8] proposed a method that uses facial features extracted from the color images. Thies *et al.* [18] proposed to generate and track a 3D face model with blendshapes using only RGB inputs. In this work, tracking is done by minimizing the difference between the appearance of the face area obtained from the input RGB image and the appearance of the two-dimensional projection of the 3D model represented by the linear sum of the source key shapes. These methods use as input only RGB images and the facial features detected with standard computer vision techniques [21] or machine learning techniques [19, 22]. The system configuration for these methods is easy, but the expression estimation fails in the case of occlusions.

More recently, the development of RGB-D cameras allowed for more robust and accurate facial re-targeting systems [12]. Several extensions have been proposed to improve the facial animation tracking accuracy [18, 20]. For example, Hsieh *et. al.* [9] proposed a method for tracking and re-targeting of expressions using blendshapes that is robust to occlusions. Concretely the authors proposed to use the RGB-D information to detect parts that occlude the face, then augment the input image with the tracked facial model to make the occluding part disappear. The facial expression tracking is then applied on the diminished image, which makes the whole system less sensitive to occlusions.

For re-targeting purposes, blendshapes are used whenever possible. Given a set of source key-shapes and target key-shapes the expression of the source model can be efficiently transferred to the target model [4, 16, 14]. The main challenge in these techniques is how to efficiently transfer the blendshape weights for realistic animation. This can be difficult when the source and target sets of blendshapes are different. Here, we propose a method to generate the target set of blendshapes that directly matches the source set of blendshapes.

Nor & Neumann et al. Carried out deformation of the mesh using the method of transferring the motion of the vertices constituting the mesh of the source to the vertices on the avatar [13]. The authors proposed to use user-defined corresponding points between the source and target mesh to transfer the expression of the source mesh to the target mesh. Although this method is specialized for transfer of facial expression change, it does not support transfer of overall mesh shape change.

Sumner et al. Performed mesh deformation by transferring polygon deformation instead of vertex movement [17]. With the help of landmark points set at dozens of corresponding positions between two meshes of source and

avatar, the mesh of the expressionless avatar is transformed into the shape of the expressionless source mesh, and then the facial expression is transferred by acquiring the correspondence information of polygons and applying the deformation of each polygon of the source to each polygon of the corresponding avatar. In this method, not only the expression part but also the movement of the whole mesh can be transferred, but as for the fine movement, the more the avatar shape differs from the source, the more accurate the result will be.

3. Preliminaries

We first define the notations used thought the paper, as well as the data structures and the 3D model representations used in our proposed method.

3.1. Notations

In this paper, vectors are denoted with bold font and small letters (*e.g.*, $\mathbf{a}$) and matrices are denoted with bold font and capital letters (*e.g.*, $\mathbf{A}$). Moreover, a list is denoted with capital letters such as L, or by small letters such as some l.

We represent a 3D mesh M with a pair of lists (V, F), where $V = \{\mathbf{v}_1, ..., \mathbf{v}_n\}$ is the list of 3D vertices and $F = \{\mathbf{f}_1, ...\mathbf{f}_m\}$ is the list of faces that connect all vertices together. Each element $\mathbf{v}_i = (x\ y\ z)^\top$ of V $(i \in [1:n])$ represents the 3D coordinates of a vertex in the reference coordinate system. Each element $\mathbf{f}_i = (\mathrm{idx}_1, ..., \mathrm{idx}_3)$ of F $(i \in [1:m])$ represents a triangular face, whose j^{th} summit is $\mathbf{v}_{\mathrm{idx}_j}$.

Each key shape of an avatar is a 3D mesh with the same list of faces and same number of vertices as the neutral mesh. Only the 3D coordinates of the vertices change between different key shapes. The set of key shapes is denoted as $\{B_i \quad (i = 0...n)\}$, where the key shape with neutral expression is B_0. Then given the set of coefficients $\{w_i \quad (i = 0...n)\}$, the blendshape model can be animated to create a new mesh M with the corresponding facial expression, where

$$M = B_0 + \sum_{i=1}^{n} \{w_i(B_i - B_0)\}. \tag{1}$$

In this work, we use two types of blendshape models: the source model and the avatar model. As shown in fig. 1, the source model is composed of a neutral expression mesh and the set of key shapes with characteristic facial expressions. On the contrary, the avatar model only has the neutral expression mesh and our objective is to generate the set of key shapes that correspond to those of the source model.

3.2. Polygon deformations

All the key shapes of a blendshape model must share the same list of faces F, while the 3D coordinates of each vertex in V change. Therefore, in order to generate the set of key shapes from a given neutral mesh, we need to deform each face of the neutral mesh towards the expected key expressions. We represent the deformation of a face with 3D affine transformations. An affine transformation consists of the 3×3 matrix $\mathbf{T}$ that represents the polygon rotation, scaling, and shear (this is the linear part); and the 3×1 vector $\mathbf{d}$ that represents the translation.

For any triangular face $\mathbf{f}_i$ $(i \in [1:m])$ with summits $(\mathrm{v}_1, \mathrm{v}_2, \mathrm{v}_3)$, we create an additional vertex $\mathbf{v}_4$ that allows to represent the polygon orientation.

$$\mathbf{v}_4 = \mathbf{v}_1 + \frac{(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)}{|(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)|}. \tag{2}$$

We can write the problem of transferring the deformation between two key-shapes of the source model to the avatar model (which is the core of the proposed method) as the problem of minimizing the following cost function with constraints.

$$\min_{(\mathbf{T}_1,\mathbf{d}_1),...,(\mathbf{T}_{|T|},\mathbf{d}_{|T|})} \sum_{i=1}^{|F|} ||\mathbf{S}_{s_i} - \mathbf{T}_{t_i}||_F^2$$
$$such\ that \quad \mathbf{T}_j\mathbf{v}_i + \mathbf{d}_j = \mathbf{T}_k\mathbf{v}_i + \mathbf{d}_k, \quad \forall i, \forall j, k \in p(\mathbf{v}_i). \tag{3}$$

Here, $\mathbf{S}$ and $\mathbf{T}$ are the linear parts of the affine transformations from the neutral shape to a given key shape for all the faces in the meshes of the source and avatar models, respectively. All linear transformations $\mathbf{S}$ are known from the source model and we aim at estimating the unknown linear transformations $\mathbf{T}$ and translations $\mathbf{d}$ of the avatar model. Note that the list of faces and number of vertices of the source and avatar models are different. As a consequence we need the correspondences (s_i, t_i) between the faces of the two models which we obtain in a semi-automatic manner as explained in section 4.2. By minimizing the difference between the affine transformations of the corresponding faces, we can accurately transfer the facial deformation from the source model to the avatar model. Note that $||\cdot||_F$ is the Frobenius norm, $|F|$ is the total number of polygons of the avatar and $p(\mathbf{v}_i)$ is the list of indices of all polygons that share the vertex $\mathbf{v}_i$.

However, depending on the shape, density and topology of the avatar mesh, solving the constrained optimization problem of equation 3 may be difficult and inefficient. As a consequence, as explained in [17] we re-write the polygon deformation optimization problem in a more efficient way. Let us consider the k^{th} face with summits $(\mathrm{v}_1, \mathrm{v}_2, \mathrm{v}_3)$ and linear transformation $\mathbf{T}_k$. Then we can write

$$\mathbf{TV} = \tilde{\mathbf{V}}, \tag{4}$$

where $\mathbf{V}$ and $\tilde{\mathbf{V}}$ are the 3×3 matrices defined as follows.

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_2 - \mathbf{v}_1 & \mathbf{v}_3 - \mathbf{v}_1 & \mathbf{v}_4 - \mathbf{v}_1, \end{bmatrix} \quad (5)$$

$$\tilde{\mathbf{V}} = \begin{bmatrix} \tilde{\mathbf{v}}_2 - \tilde{\mathbf{v}}_1 & \tilde{\mathbf{v}}_3 - \tilde{\mathbf{v}}_1 & \tilde{\mathbf{v}}_4 - \tilde{\mathbf{v}}_1, \end{bmatrix} \quad (6)$$

with $\tilde{\mathbf{v}}_i = \mathbf{T}\mathbf{v}_i + \mathbf{d}$.

From these equations, the linear transformation component $\mathbf{T}$ of a polygon's affine deformation can be expressed as a function of the vertices coordinates.

$$\mathbf{T} = \tilde{\mathbf{V}}\mathbf{V}^{-1}. \quad (7)$$

As a consequence, the constrained optimisation system in equation 3 can now be written as the following unconstrained optimisation system with respect to the new variables $\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n$.

$$\min_{\tilde{\mathbf{v}}_1,...,\tilde{\mathbf{v}}_n} \sum_{i=1}^{|F|} ||\mathbf{S}_{s_i} - \mathbf{T}_{t_i}||_F^2. \quad (8)$$

By moving the positions of the vertices of the avatar mesh so that components of the affine transformations of all faces become close to the those of the source mesh, the avatar is deformed while maintaining the adjacency and the local shape of the mesh.

Hereafter, we use this vertex-based approach to define the evaluation function for the mesh deformation.

4. Proposed method

We propose a method that generates avatar key shapes automatically from any avatar mesh with a neutral facial expression. Our proposed deformation transfer method extends the technique proposed by Sumner et al. [17] by considering the motion amplitude and local consistency of the facial landmarks. Our proposed method allows to transfer facial expressions to any avatar, even when the 3D mesh differs greatly in shape and topology compared with the template model used for tracking.

4.1. Overview

Our proposed method generates any avatar's blendshapes given only the neutral mesh and some facial landmarks provided manually by the user. We propose a mesh deformation transfer technique that keeps motion amplitude consistency to generate key-shapes of the avatar mesh similar to those of a source human template model. Our proposed method can be separated into two main steps as shown in fig. 2

In the first step, we compute the face correspondences between the (neutral) source and target meshes. To do so we first re-scale the avatar mesh to match the scale of the source mesh, and then deform the avatar mesh into the source mesh

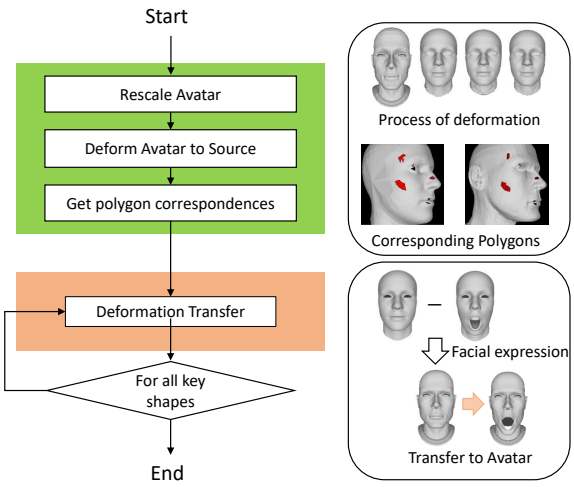

Figure 2: The pipeline of our proposed method

while keeping the mesh organization (based on facial landmarks). A fine 3D registration allows to overlay the source and warped target mesh to identify faces correspondences.

In the second step, we transfer the deformations of the source mesh from the neutral shape into all other key-shapes to the avatar model. Firstly, the linear component of the affine transformation (*i.e.*, rotation, scaling and sheering) of each face of the source model is transferred to the corresponding faces of the avatar model. Secondly the movements of the vertices define the translation part. This allows to identify the affine transformation of each face of the avatar model and generate all the key-shapes.

4.2. Faces correspondences

In this section we describe our proposed method to robustly and precisely identify the face correspondences between the source and target neutral mesh (as illustrated in fig. 2). We also describe the representation of the linear component of the affine transformations using vertices coordinates, which is a crucial point of our proposed method.

4.2.1 Re-scaling

The absolute scale of the avatar mesh is likely to be different from the one of the source mesh, which precludes any direct alignment of the two meshes. Therefore, as a preprocessing stage, we re-scale the avatar mesh to match the scale of the source mesh. Concretely, each vertex $\mathbf{v}$ of the avatar mesh is re-scaled to the new position $\mathbf{v}'$, where

$$\mathbf{v}' = \mathbf{v}\frac{x^s + y^s + z^s}{x^t + y^t + z^t}. \quad (9)$$

Here, x^s, y^s, z^s and x^t, y^t, z^t are the size of the bounding boxes in all axis coordinates of the source and avatar

meshes, respectively.

4.2.2 Non-rigid deformation

To get the correspondences between the faces of the source and avatar meshes, the avatar mesh is firstly deformed into the source mesh. To do so, we solve a constrained minimization problem with the vertices of the transformed avatar as variables. The cost function to minimize is a weighted linear sum of the following three cost functions.

The first cost function is expressed as follows.

$$E_S(\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n) = \sum_{i=1}^{|F|} \sum_{j \in adj(i)} ||\mathbf{T}_i - \mathbf{T}_j||_F^2, \tag{10}$$

where $\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n$ are the vertices of the avatar mesh after transformation, $adj(i)$ is the set of adjacent faces of the i^{th} face in the mesh. Also $\mathbf{T}_i$ is the linear component of the affine transformation of the i^{th} face. Note that E_S is a function of the vertices of the mesh as discussed in sec. 3.2.

The second cost function is expressed as follows.

$$E_I(\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n) = \sum_{i=1}^{|F|} ||\mathbf{T}_i - \mathbf{I}||_F^2, \tag{11}$$

where $\mathbf{I}$ is the 3×3 identity matrix.

The third cost function is expressed as follows.

$$E_C(\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n) = \sum_{i=1}^{n} ||\mathbf{v}_i - \mathbf{c}_i||^2, \tag{12}$$

where $\mathbf{c}_i$ is the closest valid point on the source mesh for the vertex $\mathbf{v}_i$ on the avatar mesh. The angle between the normal vectors of $\mathbf{v}_i$ and $\mathbf{c}_i$ is enforced to be within $90°$. As illustrated in fig. 3, by considering this angle, it becomes possible to differentiate between topologically far regions such as points in the upper and lower lips.

By minimizing the weighted linear sum of the above three cost functions, the vertices coordinates of the avatar mesh after deformation $\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n$ can be estimated. The constrained optimization problem to solve is expressed as follows.

$$\begin{aligned} &\min_{\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n} E(\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n) = w_s E_S + w_i E_I + w_c E_C \\ &\quad such\ that \quad \tilde{\mathbf{v}}_{corr(i)} = [\mathbf{v}_i]_{s,neut}, \quad i \in L, \end{aligned} \tag{13}$$

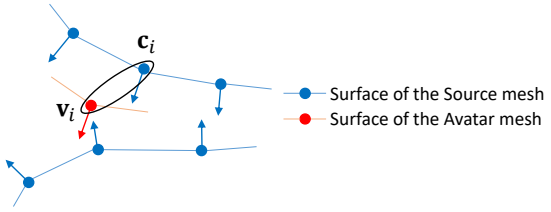

Figure 3: Illustration of closest valid point identification

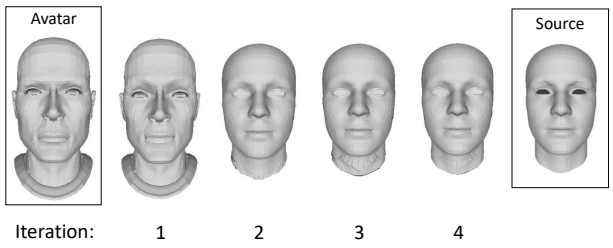

Figure 4: Non-rigid deformation of the avatar mesh towards the source mesh

where $[\mathbf{v}]_{s.neut}$ indicates that the vertex $\mathbf{v}$ is a vertex on the source neutral mesh, L is a list of facial landmark indices on the source mesh (as described below), $corr(i)$ is index of the i^{th} landmark of the avatar mesh corresponding to the i^{th} landmark of the source mesh

In order to minimize equation 13, a set of facial landmarks provided by the user is required. We use 51 facial landmarks around the eyes, eyebrows, nose and mouth as in dlib [1]. The source mesh is already given a fixed set of landmarks L. Then we develop a user interface for the user to easily select the corresponding 51 facial landmarks on the avatar mesh. The important point here is that the landmarks on the avatar mesh must have the same semantic meaning as those of the source mesh (*e.g.*, "left corner of the eye" or "middle of upper lip"). This is crucial to help good convergence of the deformation of the avatar neutral mesh towards the source neutral mesh. If the resolution of the avatar mesh is insufficient, we up-sample the avatar mesh. Then, we can always select the corresponding landmarks at the semantically correct position.

In order to obtain optimal results, the minimization of the system in equation 13 is repeated several times while changing the weights applied to each cost function. In the experiments, we used four iterations, with $w_s = 1, w_i = 0.1, w_c = 0$ for the first iteration and $w_s = 1, w_i = 0.001, w_c = 1$ for the subsequent iterations. Figure 4 illustrates the non-rigid deformation process.

4.2.3 Matching

With the transformed avatar mesh it becomes possible to identify the faces correspondences. We align the transformed avatar mesh to the source mesh and select the correspondences as the nearest faces. Figure 5 illustrates the face correspondences acquisition.

Firstly, we compute the centroids and normal vectors of all the faces of the transformed avatar mesh. Secondly, all the faces of the avatar mesh are matched to the closest face on the source mesh that satisfies the constraint that the angle

[1] https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

between the normal vectors is smaller than $90°$. The same process is done in the reverse way from the source mesh to the avatar mesh. In this way, we obtain one-to-many and many-to-one correspondences between the faces of the target and source meshes. As a consequence, meaningful correspondences can by obtained even between meshes having totally different density. The correspondence list is expressed as follows.

$$M = \{(s_1, t_1), (s_2, t_2), ..., (s_{|M|}, t_{|M|})\}, \quad (14)$$

where s_i is the index of the face in the source mesh and t_i is the index of the face in the avatar mesh.

4.3. Deformation transfer

We use the face correspondences between the source and avatar meshes to generate a set of key-shapes of the avatar model with the same facial expressions as the source blendhsape model. To this end we transfer the deformation of the source mesh from the neutral expression to its various expressions to the neutral avatar mesh. This deformation transfer is guided by the facial landmarks.

4.3.1 Polygon deformation transfer

We use the following displacement cost function to transfer the deformation of a face in the source mesh to its corresponding face in the avatar mesh.

$$E_d(\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n) = \sum_{i=1}^{|F|} ||\mathbf{S}_{s_i} - \mathbf{T}_{t_i}||_F^2, \quad (15)$$

which is similar to equation 8 with using the list of face correspondences $M = \{(s_1, t_1), (s_2, t_2), ..., (s_{|M|}, t_{|M|})\}$.

In the original work of Sumner *et.al.* [17], only this displacement cost function is used to guide the deformation. We propose to use two additional cost functions to achieve more faithful deformation transfer.

4.3.2 Landmark guided deformation transfer

In order to guarantee the successful expression transfer even to meshes that cannot be handled only by the linear transformation component, we propose to use the motion of the manually selected facial landmarks. We thus define the following landmark cost function.

$$E_l(\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n) = \sum_{k=1}^{6} \sum_{i \in L_k} ||\mathbf{A}_k \mathbf{R}_k \mathbf{u}_i - \tilde{\mathbf{u}}_{corr(i)}||^2, \quad (16)$$

$$\text{where} \quad \begin{cases} \mathbf{u}_i = [\mathbf{v}_i]_{s.exp} - [\mathbf{v}_i]_{s.neut} \\ \tilde{\mathbf{u}}_{corr(i)} = \tilde{\mathbf{v}}_{corr(i)} - [\mathbf{v}_{corr(i)}]_{t.neut} \end{cases}$$

Here, L_k is the list of indices of the landmarks on the source mesh. $corr(i)$ is the index of the landmark in the avatar mesh that corresponds to the i^{th} landmark on the source mesh. $\mathbf{u}_i$ and $\tilde{\mathbf{u}}_{corr(i)}$ are the displacement vectors of the i^{th} landmark between the neutral expression and the current expression for the source and avatar meshes, respectively. $[\mathbf{v}]_{s.exp}, [\mathbf{v}]_{s.neut}, [\mathbf{v}]_{t.neut}$ indicate that $\mathbf{v}$ is a point on the source mesh with expression, on the neutral source mesh and on the neutral avatar mesh, respectively. Finally, $\mathbf{A}_k$ and $\mathbf{R}_k$ are 3×3 matrices that allow to adjust the amount and direction of the movement of the landmarks.

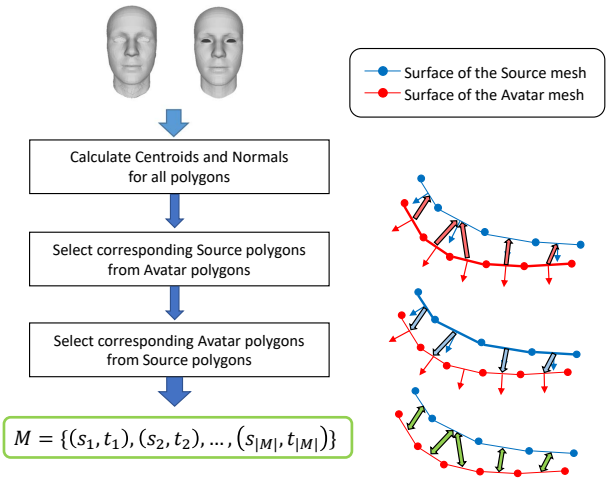

Figure 5: Face correspondences acquisition pipeline

The size and orientation of the facial parts (like the nose for example) vary greatly depending on the chosen avatar mesh (for example if the avatar is a horse, the shape is totally different than the face of a human). Therefore, we propose to divide the facial parts of the source and avatar models into six parts: right eyebrow, left eyebrow, nose, right eye, left eye and mouth. We measure the size and orientation of each facial part and coherently adjust the amount and direction of the landmarks motion.

Firstly, we measure the orientation of each facial part. We organise the landmarks into the different parts in the list $L_k (k \in 1...6)$. From each group we select the landmarks that are best suited to compute the facial part's orientation, and we denote as $L_k^{'}$ the list made of these landmarks. We then minimize the following cost function so that the normal vector $\mathbf{n}_k$ becomes approximately perpendicular to all lines that can be drawn between the landmarks in $L_k^{'}$.

$$E(\mathbf{n}_k) = \sum_{i=0}^{|L_k^{'}|} \sum_{j=i}^{|L_k^{'}|} \mathbf{n}_k \cdot (\mathbf{v}_{L_k^{'}[i]} - \mathbf{v}_{L_k^{'}[j]}), \quad k \in [1:6] \quad (17)$$

From the orientation vector $\mathbf{n}_k$ of the k^{th} part of the face, we compute the rotation matrix $\mathbf{R}_{k,s}, \mathbf{R}_{k,t}$ that aligns $\mathbf{n}_k$ with the z axis. Those rotation matrices are computed independently for each of the 6 facial parts.

To measure the size of each facial part, we first use the rotation matrices $\mathbf{R}_{k,s}$ and $\mathbf{R}_{k,t}$ to align the facial part with the z axis. Then, we compute the 3D bounding box of the aligned facial parts for the source and avatar meshes with size $a_{x,s}\ a_{y,s}\ a_{z,s}$ and $a_{x,t}\ a_{y,t}\ a_{z,t}$, respectively. This allows us to obtain the size ratio $\mathbf{A}_k$ as follows.

$$\mathbf{A}_k = \begin{bmatrix} \frac{a_{x,t}}{a_{x,s}} & 0 & 0 \\ 0 & \frac{a_{y,t}}{a_{y,s}} & 0 \\ 0 & 0 & \frac{a_{z,t}}{a_{z,s}}. \end{bmatrix} \quad (18)$$

In the case of the mouth part (for which the range of movements cannot be measured by only the neutral avatar mesh) we compute the ratio in the y direction using the distance between the lower end of the nose and the center of the lower lip. The ratio in the z direction is set to 1.

We can then faithfully transfer the motion of the landmarks from the source model to the avatar model by warping the landmarks of the source mesh to the re-sized and aligned coordinate system of the corresponding face part in the avatar mesh. The landmark motion vector $\tilde{\mathbf{u}}_{corr(i)}$ in the avatar mesh is obtained as the transformed motion vector $\mathbf{A}_k\mathbf{R}_k\mathbf{u}_i$ of the corresponding landmark in the source mesh.

4.3.3 Optimization

We compute the new vertices coordinates $\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n$ of the deformed avatar mesh by minimizing the following cost function that is a weighted sum of the cost functions E_S and E_I introduced in above.

$$\min_{\tilde{\mathbf{v}}_1,...,\tilde{\mathbf{v}}_n} E(\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_n) = w_d E_d + w_l E_l + w_i E_I \quad (19)$$

In our experiments, we set $w_d = 1$, $w_l = 100$, $w_s = 10$ and $w_i = 1$. We repeat the minimization process for all the key-shapes of the source model to obtain a blendshape avatar model that corresponds to the source model.

5. Experiments

We evaluate the ability of our proposed method to generate faithful blendshape models of avatars by using several meshes with totally different shapes (like horse and cat). We compared the results obtained with our proposed method with those obtained with the state-of-the-art mesh deformation transfer method of Sumner *et.al.*, [17] and we show the results in figures 6, 7, 8 and 9[2].

For each avatar, the selected landmarks are shown in the lower left corner of the corresponding figure. We use landmarks only on the deformable parts of the human face, which is enough to re-target the facial expression (no landmarks on the ear for example). While selecting the landmarks for human-like avatars such as the "old man" can be done precisely, we can only set approximate positions of the landmarks for avatars with totally different shapes.

[2]Video available at http://limu.ait.kyushu-u.ac.jp/e/member/member0042.html

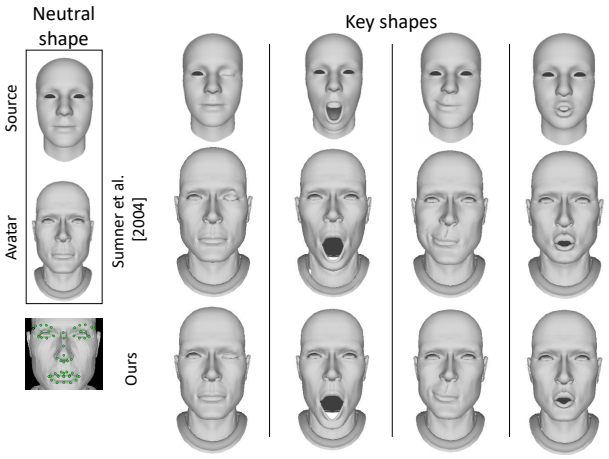

Figure 6: Comparative results obtain with "old man".

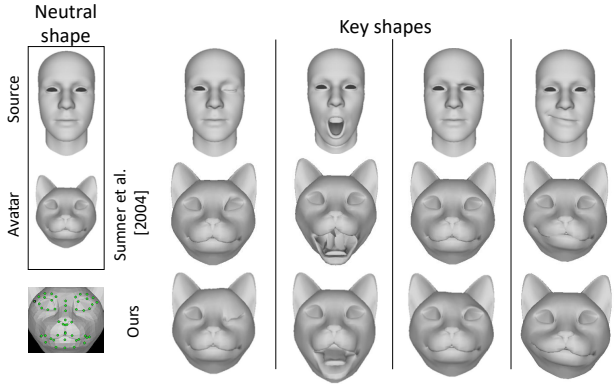

Figure 7: Comparative results obtain with "cat".

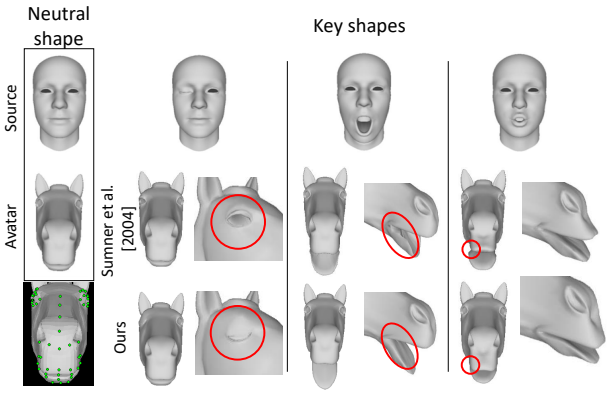

Figure 8: Comparative results obtain with "horse".

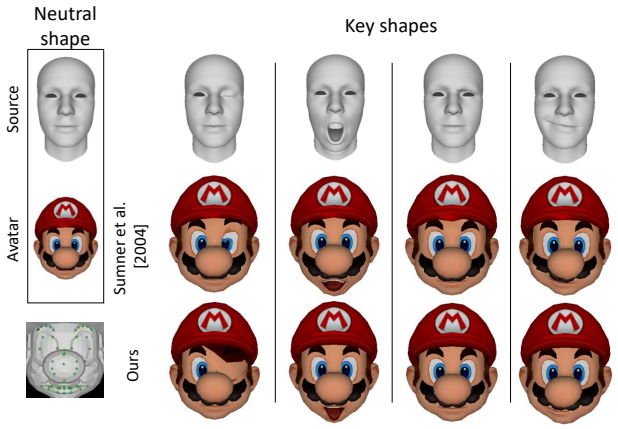

Figure 9: Comparative results obtain with "Mario".

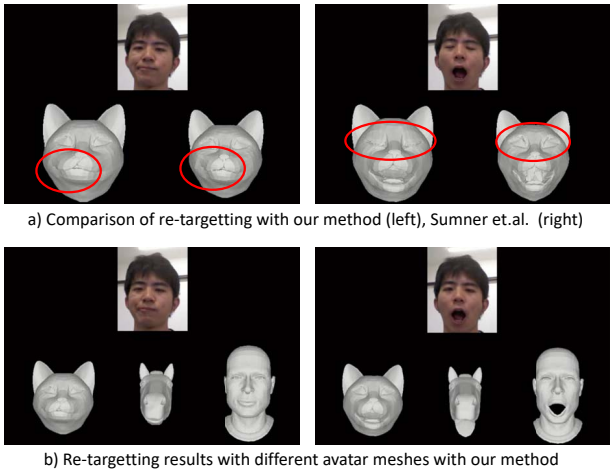

Figure 10: Re-targeting results.

Figure 6 shows the results obtained with the human-like avatar "old man" when using our proposed method and the one in [17]. In this case, accurate face correspondences could be obtained, which allowed to generate faithful blendshape models with both methods. However, as we can see from figures 7 and 8 the method proposed by Sumner *et.al.*, [17] could not generate correct facial expression for avatar meshes with totally different shape than the source mesh. The main problems arose around the eye region. This is because perfect face correspondences cannot be obtained in these cases. Nevertheless, our proposed method could successfully generate the correct facial expressions even in these challenging situations by using landmark deformations to guide the deformation transfer.

Figure 9 shows the comparative results obtained with the avatar "Mario". This case is extremely challenging because not only the density of the source and avatar mesh are totally different, but the topology of meshes are also different (the moustaches and the hat for example). The low density of faces in the avatar mesh explains why when the eye gets closed, part of the forehead moves unnaturally. One possible solution to avoid this problem would be to up-sample the avatar mesh to increase the face density. Moreover some parts of the face may penetrate other parts of the face in an unnatural way (like the teeth or the mustache). In order to cope with avatars composed of a plurality of meshes and avatars having many irregularities, further consistency constraints should be considered, which is left for future works.

We demonstrate the potential of our proposed method with a concrete re-targeting application. We used the facial expression tracking method proposed by Thomas *et.al.*, [20] to compute in real-time the blendshape coefficients of the source template model, using a Kinect V1 RGB-D camera. Then we re-targeted these expressions to the avatar meshes with the key-shape created with our proposed method. Figure 10 a) compares the results obtained with our proposed method (on the bottom left of the screen) with the results obtained when using the state-of-the-art mesh deformation method proposed by Sumner *et.al.*, [17] to create the key-shapes (on the bottom right). As we can see from these results, our proposed method allowed us to obtain much more faithful expression re-targeting results. The advantage is particularly clear in the eye area (right side of fig. 10 a)). Only our proposed method could correctly transfer the eye shutter motion, which is a very important facial expression. In addition, fig. 10 b) shows that our proposed method can be applied to various kinds of avatar meshes, with similar results (the video of this experiments is available in the supplemental materials). Note that by creating the key-shapes that match the input source model, simple re-targeting approach by directly applying the tracked blendshapes coefficients becomes possible.

6. Conclusion

In this paper, we proposed a method to generate the key-shapes of an avatar mesh in a semi-automatic manner by transferring the deformations of a source blendshape model. We extend the method proposed by Sumner *et.al.*, [17] by adding a landmark-guided deformation constraint. By doing so, faithful facial expressions of the avatar could be easily generated. Our experimental results confirmed the advantage of our proposed method over the state-of-the-art and demonstrate the potential of this technology for realistic re-targeting purpose. In our future work, we plan to add additional constraints to avoid unnatural intersection of different parts of the face. We will also extend the method to transfer fine facial details such as wrinkles.

References

[1] I. Baran, D. Vlasic, E. Grinspun, and J. Popović. Semantic deformation transfer. In *ACM Transactions on Graphics (TOG)*, volume 28, page 36. ACM, 2009. 2

[2] T. Beeler, F. Hahn, D. Bradley, B. Bickel, P. Beardsley, C. Gotsman, R. W. Sumner, and M. Gross. High-quality passive facial performance capture using anchor frames. In *ACM Transactions on Graphics (TOG)*, volume 30, page 75. ACM, 2011. 2

[3] M. Ben-Chen, O. Weber, and C. Gotsman. Spatial deformation transfer. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 67–74. ACM, 2009. 2

[4] S. Bouaziz, Y. Wang, and M. Pauly. Online modeling for realtime facial animation. *ACM Transactions on Graphics (ToG)*, 32(4):40, 2013. 2

[5] D. Bradley, W. Heidrich, T. Popa, and A. Sheffer. High resolution passive facial performance capture. In *ACM transactions on graphics (TOG)*, volume 29, page 41. ACM, 2010. 2

[6] E. Chuang and C. Bregler. Performance driven facial animation using blendshape interpolation. *Computer Science Technical Report, Stanford University*, 2(2):3, 2002. 2

[7] C. Curio, M. Breidt, M. Kleiner, Q. C. Vuong, M. A. Giese, and H. H. Bülthoff. Semantic 3d motion retargeting for facial animation. In *Proceedings of the 3rd symposium on Applied perception in graphics and visualization*, pages 77–84. ACM, 2006. 2

[8] L. Dutreve, A. Meyer, and S. Bouakaz. Feature points based facial animation retargeting. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pages 197–200. ACM, 2008. 2

[9] P. L. Hsieh, C. Ma, J. Yu, and H. Li. Unconstrained realtime facial performance capture. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1675–1683, June 2015. 2

[10] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. Pighin, and Z. Deng. Practice and Theory of Blendshape Facial Models. In S. Lefebvre and M. Spagnuolo, editors, *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014. 2

[11] H. Li, T. Weise, and M. Pauly. Example-based facial rigging. In *Acm transactions on graphics (tog)*, volume 29, page 32. ACM, 2010. 2

[12] H. Li, J. Yu, Y. Ye, and C. Bregler. Realtime facial animation with on-the-fly correctives. *ACM Trans. Graph.*, 32(4):42–1, 2013. 1, 2

[13] J.-y. Noh and U. Neumann. Expression cloning. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM. 2

[14] Y. Seol, J. Seo, P. H. Kim, J. Lewis, and J. Noh. Artist friendly facial animation retargeting. In *ACM Transactions on Graphics (TOG)*, volume 30, page 162. ACM, 2011. 2

[15] E. Sifakis, I. Neverov, and R. Fedkiw. Automatic determination of facial muscle activations from sparse motion capture marker data. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 417–425, New York, NY, USA, 2005. ACM. 2

[16] J. Song, B. Choi, Y. Seol, and J. Noh. Characteristic facial retargeting. *Computer Animation and Virtual Worlds*, 22(2-3):187–194, 2011. 2

[17] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 399–405, New York, NY, USA, 2004. ACM. 2, 3, 4, 6, 7, 8

[18] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. Face2Face: Real-time Face Capture and Reenactment of RGB Videos. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016. 1, 2

[19] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2387–2395, 2016. 2

[20] D. Thomas and R.-I. Taniguchi. Augmented blendshapes for real-time simultaneous 3d head modeling and facial motion capture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3299–3308, 2016. 2, 8

[21] M. Uřičář, V. Franc, D. Thomas, A. Sugimoto, and V. Hlaváč. Real-time multi-view facial landmark detector learned by the structured output svm. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 2, pages 1–8. IEEE, 2015. 1, 2

[22] Y. Wu, T. Hassner, K. Kim, G. Medioni, and P. Natarajan. Facial landmark detection with tweaked convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):3067–3074, 2017. 1, 2