2D-to-3D Facial Expression Transfer

Gemma Rotger*, Felipe Lumbreras*, Francesc Moreno-Noguer† and Antonio Agudo†
*Computer Vision Center and Dpt. Ciències de la Computació, Universitat Autònoma de Barcelona, Spain
†Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain

***Abstract*—Automatically changing the expression and physical features of a face from an input image is a topic that has been traditionally tackled in a 2D domain. In this paper, we bring this problem to 3D and propose a framework that given an input RGB video of a human face under a neutral expression, initially computes his/her 3D shape and then performs a transfer to a new and potentially non-observed expression. For this purpose, we parameterize the rest shape –obtained from standard factorization approaches over the input video– using a triangular mesh which is further clustered into larger macro-segments. The expression transfer problem is then posed as a direct mapping between this shape and a source shape, such as the blend shapes of an off-the-shelf 3D dataset of human facial expressions. The mapping is resolved to be geometrically consistent between 3D models by requiring points in specific regions to map on semantic equivalent regions. We validate the approach on several synthetic and real examples of input faces that largely differ from the source shapes, yielding very realistic expression transfers even in cases with topology changes, such as a synthetic video sequence of a single-eyed cyclops.**

I. Introduction

The acquisition of time-varying 3D face models has become an increasingly popular technology for transferring real human deformations to virtual avatars in movies and video games. Early motion capture systems were based on markers placed over the face which could capture sparse representations. However, besides being intrusive, these methods could not acquire accurate dense reconstructions, and as a consequence, small details of subtle gestures are not recovered. Very recently, dense face expression transfer has been addressed from a Deep Learning (DL) perspective, by learning 2D-to-2D mappings from large amounts of training images [1], [2].

In this paper, we move beyond these approaches and provide a solution to densely transfer face expressions in the 3D domain, which exploits the potential of the so-called structure-from-motion algorithms and does not require training data at all nor placing markers on the actor's face. More specifically, given an input RGB video of a person under a neutral or changing expression, we first leverage on a standard structure from motion framework to estimate his/her 3D face shape. On the other hand, we assume a 3D source expression is provided for transferring, given a low-rank 3D dataset of expressions, composed by a rest shape and a number of blending shapes under specific expressions for synthetic validation, or by means of real-world facial expressions from a video. We then develop an approach to map the initial 3D shape of our actor's face to the shape at rest of the source actor. This 3D-to-3D mapping requires addressing several challenges, including different topologies, different mesh resolutions and noise in the input shape. We tackle these challenges by identifying common semantic regions into the input and target faces and locally solving the mapping for each of these regions. Once this mapping is resolved we are then able to transfer a wide spectrum of 3D expressions to the input actor's face. See the overview of the approach in Fig. 1.

We extensively evaluate and demonstrate the effectiveness of our pipeline on a wide number of synthetic and real examples, even considering cases with large topology changes between the input and the source shapes. For instance, we show that the expressions of our low-rank 3D dataset can be transferred to a single-eyed face of a cyclops.

II. Related Work

Facial performance capture has been extensively studied during the past years [3] [4] [5]. The most standard approach to address this problem is the use of facial markers [6] or light patterns [7] to simplify the tracking process. Besides being intrusive, the sparsity of the methods prevents the capture of high deformation details like expression wrinkles and skin folds. Otherwise, markerless approaches present a non-intrusive alternative, but tracking may fail due to fast motion or complex deformations. This type of methods includes linear [8], or multilinear [9], [10] models, multiview-stereo solutions [11], [12], and techniques based on RGB-D data [13], [14]. Most previous approaches have been focused on recovering a small set of parameters, and consequently, they are not suitable to retrieve detailed face deformations.

Impressive results have been achieved in dense 2D-to-2D facial expression transfer [14], [15], and 3D-to-3D [16]. In this context, DL approaches have been firstly introduced in this area [17] with prominent image-to-image results. Very recently, a 2D-to-3D approach shows striking results [18], but due to the nature of the formulation, it is unable to retrieve fine-details, and its applicability is limited to the expressions lying in a linear shape subspace with known rank. We solve previous limitations, by removing the need for expression and the identification of a large amount of training data.

III. Our Approach

Let us consider a set of b 3D expressions together with their corresponding shape at rest. An arbitrary face shape is made of p 3D points, represented by the matrix $\mathbf{S}_S(\alpha) = [\mathbf{s}_1, \ldots, \mathbf{s}_i, \ldots, \mathbf{s}_p]$, where the i-th column includes the 3D coordinates for the point $\mathbf{s}_i \in \mathbb{R}^3$. The parameter $\alpha = \{0, \ldots, b\}$ denotes a basis index, being $\alpha = 0$ for the shape at rest. This geometric representation can also be described by using

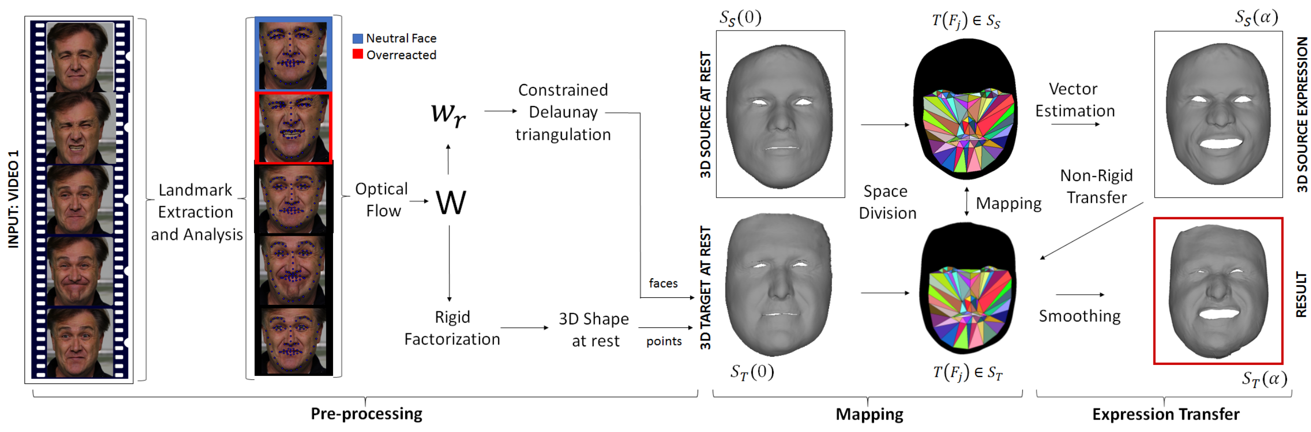

Fig. 1: **Overview of our expression-transfer approach**. Our approach consists of three stages: video to 3D shape with landmarks, mapping, and finally an expression transfer with smoothing. In the first stage (see the left part in the picture), a 3D shape from the optical flow is computed by rigid factorization, where overreacted measurements can be automatically removed to guarantee convergence, and a reference frame is selected. After that, we perform a sub-region mapping stage (represented in the middle), considering both resting target and the corresponding source face. In both cases, the shape is split into subregions to define a 3D-to-3D mapping between surfaces, and fitting the model. Finally, we perform the expression transfer stage (see right part) where the 3D configuration of a specific expression is transferred from the 3D source to the 3D target face model.

a mesh of T triangular patches, where each vertex corresponds to a 3D surface point. In a similar manner, we can define now a target face shape composed of n 3D points, represented by the matrix $\mathbf{S}_T(\alpha) = [\mathbf{s}_1, \ldots, \mathbf{s}_i, \ldots, \mathbf{s}_n]$, which can be discretized into R triangular elements. Again, the parameter α can take a value from 0 to b, booking the entry of 0 to denote our input estimation.

In this paper, given a monocular video of a potentially non-rigid face, our goal is to estimate its 3D reconstruction in a specific and non-observed expression. Without loss of generality, we consider initially the specific expression is included in the source actor expression set, and its selection is assumed to be known. In addition, our approach can be also applied for on-line estimation, where an arbitrary facial expression of a source avatar is acquired and then transferred to our estimated 3D target model. The challenge of this work is to automatically estimate the 2D-to-3D mapping between the observed face acquired in the video, and the target face with a specific expression without requiring any training data at all to constrain the solution, nor the specification of a pre-defined shape basis to project the solution. It is worth pointing out that our expression-transfer algorithm can even work when the resolution of both shapes $\mathbf{S}_S(\alpha)$ and $\mathbf{S}_T(\alpha)$ are different, i.e., when the number of points and triangular faces differs.

To this end, we propose an efficient algorithm which works in a three consecutive stages: an extended structure-from-motion stage, to obtain a 3D estimation from 2D that considers both sparse and dense solutions; a stage to establish the mapping between the target and the source shape; and finally, an expression-transfer stage that allows recovering the 3D estimation with a specific expression. A summary of our approach is shown in Fig. 1. Next, we explain in deeper every stage of our approach.

IV. From RGB video to 3D Model

In the first stage, our goal is to recover a 3D shape model solely from a monocular video. A priori, no information about the video is assumed and hence the observed face could potentially undergo non-rigid motions. In the last decade, this problem has been addressed by non-rigid structure from motion approaches [19], [20], [21], showing accurate results even with incomplete 2D point tracks. However, our approach only requires a 3D model or rest shape, rather than knowing a 3D shape per image frame, which can be easily computed by rigid structure from motion techniques [22].

In order to automatically retrieve the 3D model, we first obtain a sparse collection $\mathcal{F}$ of 68 landmarks by applying OpenFace [23] over the monocular video. These observations are then used to find the image frames where the neutral expressions appear, assigning the most neutral expression as the reference frame. To this end, we perform a Procrustes analysis of every frame against a 2D model extracted from our source $\mathbf{S}_S(0)$ and determining which frames are neutral in terms of deformation or those that are overreacted and may make the process fail. We select those with an error lower than a threshold and discard those with a larger value, taking the frame as the one with the lowest error. Finding this frame is a key factor in our approach, since over this frame we compute a facial mask, and establish a reference to estimate optical flow.

After that, we apply state-of-the-art dense optical flow [24] to obtain 2D trajectories in the monocular video, collecting all observations in the measurement matrix $\mathbf{W} \in \mathbb{R}^{2f \times n}$, where f is the number of frames and n the number of points. Finally, we have to infer a 3D model from the measurement matrix $\mathbf{W}$. In general terms, we could apply recent works on non-rigid reconstruction and exploit the estimated time-varying shape to compute a mean shape. However, these algorithms can become computationally demanding for dense scenarios. To solve this limitation, we rely on rigid [22] approaches, since it is well-known the 3D estimation these methods produce is accurate enough when the motion is rigid dominant, as it is the case for face acquisition. We have observed experimentally the mean shape by applying rigid factorization is roughly the same as that estimated with non-rigid approaches, showing robustness on the estimations.

For later computations, we also obtain the point connectivity over the 2D reference frame, where the connection is easily defined, by means of a Delaunay triangulation [25]. For simplicity, we have used the Delaunay triangulation, although we could take advantage of having an estimation of the 3D rest shape and easily use alternative connectivity algorithms.

V. Mapping function

In the second stage, our goal is to establish a mapping function between the source and target models, respectively. Recall that this problem is not direct since the number of points and triangles to define every shape can be different.

To establish a correspondence, we divide every face shape by means of 101 pieces (some pieces can be seen in Fig. 1), grouping the points with a similar topology. These pieces are defined by $T(\mathcal{F})$ where T denotes a super-triangular piece defined by three points of $\mathcal{F}$. Doing this, we can simplify the process of comparing faces of different topologies (races, genders, etc.) whose geometric information is also different.

Our model $\mathcal{M}$ is unsupervised and is able to classify each point on the 3D mesh to their corresponding subregion. A second classification step includes an iterative KNN to correct all the points that cannot be labeled in the first step. The iterative KNN labels the dissident points according to the values of their direct neighbors, but to prevent the shape change in the groups, we do not label a point if it has not at least K direct neighbors assigned. With this condition, we will iterate until enough reliable information is available to label the points. Lastly, we define the mapping model in which we estimate the point correspondence between the two geometries by generalized barycentric parametrization techniques.

Point subregion classification. Our model is a parametric mapping function defined by:

$$\mathcal{M} : \mathbf{S}_S(\mathcal{P}) \mapsto \mathbf{S}_T(\mathcal{P}), \tag{1}$$

where $\mathcal{P}$ represents a shape parametrization, and $\mathbf{S}_S(\mathcal{P})$ is known for the full set of $\mathcal{P}$, while $\mathbf{S}_T(\mathcal{P})$ is only known at the rest parametrization $\mathbf{S}_T(0)$.

We compute the 2D projection over the XY plane of the 3D points $\mathbf{S}_T(0)$ and the triangles $T(\mathcal{F})$. Then, we test for all the possible pieces $T(\mathcal{F})$ and all the points in $\mathbf{S}_S(0)$, if a point $v_t^i \in \mathbf{S}_T(0)$ lies in any triangle. To find out if v_t^i lies in the triangle $T(\mathcal{F}^j)$ we define an inside-outside function which is defined as:

$$\mu(v_t^i, t_j^n) = \begin{cases} 1, & \text{if } \mathbf{N}((t_j^{n+1} - t_j^n) \times (v_t^i - t_j^n)) \geq 0 \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$

where t_j^0, t_j^1 and t_j^2 represent the vertices of the triangle $T(\mathcal{F}^j)$, and $\mathbf{N}$ is the corresponding normal vector.

If the sum of the function $\mu(v_t^i, t_j^n)$ for every pair of vertices of the triangle is equal to 3, the point v_t^i lies inside the triangle $T(\mathcal{F}^j)$ and can be defined by barycentric coordinates. Finally, we have a set of point labels to one of the 101 different patches and its corresponding barycentric coordinates Λ_T. We do the same proceeding with triangular elements on the template Λ_S. In this case, an element can belong to more than one subregion, for example, if a triangle has each of its vertices on different subregions the triangle will belong to all these regions, so by its geometry, it can belong to three subregions.

Due to the precision of computation, we need to correct some points which remain unassigned. We resolve this ambiguity with an iterative clustering process using K-Nearest Neighbors (KNN), in which we assign to every unassigned point the most plausible value rely upon their KNN. It must be iterative due to the configuration of the clusters which are triangles instead of ellipses. So for each iteration of the KNN, we just assign a point if at least it has K labeled direct neighbors. If not, the point remains unassigned until enough reliable information is available, thus the triangular shape of the clusters remains intact.

Model Fitting. In addition to subregion parametrization of 3D points, we can achieve a finer definition over a smaller element of $\mathbf{S}_S(0)$. Let C_S be the set of all the elements of $\mathbf{S}_S(0)$ where c_S^k represents a single element of C_S. By the previous step, we can know that an element belongs to a subregion (or more), so $c_S^k \in T(\mathcal{F}_|)$. As we mentioned previously, we can represent the 3D point in barycentric coordinates, so now we have $\lambda_t^i \equiv v_t^i$. If we project the point v_t^i over each plane $C_S(\mathcal{F}_|)$ we minimize the distance of the point over the plane projection $d = \mathbf{N}(v_i - t_j^n)$, subject to the condition that the projection lies in the triangle ($v_i^s \in c_S^k$).

The previous parametrization of the point allows approximating it to the parametric solution, $\mathbf{T}^{\mathcal{F}_|}$ and $\mathbf{T}^{C_T^k}$ encode the edges of the triangles $T(\mathcal{F}^i)$ and C_T^k, respectively:

$$\mathbf{T}^{\mathcal{F}_|} = \begin{bmatrix} t_{x0}^j - t_{x2}^j & t_{x1}^j - t_{x2}^j \\ t_{y0}^j - t_{y2}^j & t_{y1}^j - t_{y2}^j \\ t_{z0}^j - t_{z2}^j & t_{z1}^j - t_{z2}^j \end{bmatrix}, \tag{3}$$

$$\mathbf{T}^{C_T^k} = \begin{bmatrix} c_{x0}^k - c_{x2}^k & c_{x1}^k - c_{x2}^k \\ c_{y0}^k - c_{y2}^k & c_{y1}^k - c_{y2}^k \\ c_{z0}^k - c_{z2}^k & c_{z1}^k - c_{z2}^k \end{bmatrix}. \tag{4}$$

Now, with Eq. (5) we can map any point from the target geometry $\mathbf{S}_T(0)$ to a point in $\mathbf{S}_S(0)$ represented in barycentric coordinates. We can see that this is not a one-to-one correspondence mapping, so it is a favorable circumstance of our method that works well with meshes with a different number of vertices or faces. $\varepsilon(T_{\mathcal{F}_|})$ represents the deformation of the triangle formed with landmarks $\mathcal{F}_|$ on $\mathbf{S}_S$ to adapt the geometry of the equivalent triangle in $\mathbf{S}_T$, and it is defined as $\varepsilon(\mathbf{T}_{\mathcal{F}_|}) = \mathbf{T}_{\mathcal{F}_|} - \mathbf{T}_{\mathcal{F}_|}^{tp}$.

Barycentric coordinates are also very useful to compute a vector impact over a point inside a triangle when information is only available at the vertices, as in our case. So we can take advantage of the current parametrization and obtain the displacement vector at the exact point:

$$\mathcal{M} : \mathbf{v}_s^{i'} = ((\mathbf{T}_{C_S^k}(\varepsilon(\mathbf{T}_{\mathcal{F}_|})^{-1}(\mathbf{v}_t^i - \mathbf{t}_3)) + \mathbf{c}_3^k) + \mathbf{d})(-\mathbf{N}), \quad (5)$$

$$\mathcal{M}^{-1} : \mathbf{v}_t^i = (\varepsilon(\mathbf{T}_{\mathcal{F}_|})(\mathbf{T}_{C_S^k}^{-1}(\mathbf{d}_i'(\mathbf{N}) - \mathbf{c}_3^k - \mathbf{d})) + \mathbf{t}_3). \quad (6)$$

VI. Expression Transfer and Smoothing

Expression Transfer. Once the vectors are computed, they can be transferred as a point using the inverse parametrization model described in Eq. (6). This mapping function allows returning the computed vectors to the initial target geometry $\mathbf{S}_T(0)$. So a point $v_R^i \in \mathbf{S}_T(P_b)$ can be computed as the addition of the mapped vector d_i to the initial point in $\mathbf{S}_T(0)$:

$$v_t^i(\mathcal{P}_b) = v_t^i(0) + d_i(\mathcal{P}_b) \; . \quad (7)$$

Smoothing energy. The central thought of the smoothing process is to provide smooth 3D expressions without missing detail. Given a transferred expression $\mathbf{S}_T(\alpha)$, the objective is to obtain a smoothed $\mathbf{S}_T^S(\alpha)$ preserving as much detail as possible. The smoothing energy presented in Eq. (8) makes use of a specified data smoothing model (Total Variation) and a Gaussian connectivity weight over the expression vectors:

$$E = \sum_{l=1}^{L} \frac{1}{\omega_l \omega_d} \sum_{i \in V} \sum_{j \in N_l} \| s_t(\alpha)^i - s_t(\alpha)^j \|_2 \; . \quad (8)$$

Our approach minimizes for every point in the region of interest $\mathbf{S}_T(\alpha)$, the distance regarding their neighbors using a Gaussian weighed Total Variation approach. The peculiarity of our smoothing functional is the use of a Gaussian weight over the l level neighbors (ω_l). Each vertex is considered neighbor of another vertex just once, and this connection is done using the minimum value geodesic distance, so given the adjacency matrix, we can evaluate for each vertex of $\mathbf{S}_T(\alpha)$ how vertices are connected. So we can define, as neighbors of level l all of those neighbors with geodesic distance equals to l which are nor included on a lower-level. So, for a vertex $\mathbf{s}_t(\alpha)^i$ with neighbors at level l, $\sum_{m=1}^{l-1} N_l \cap N_m = \varnothing$. The facial landmarks do not need to be estimated as they have a direct linear mapping, so they neither need to be smoothed.

VII. Experimental Results

We now present our experimental evaluation for different types of datasets on both synthetic and real videos (see videos in the supplemental material). For every dataset, we indicate by (V/F) the number of vertices and triangular faces, respectively. We consider the face synthetic sequences denoted as Seq3 [26] (28,887/57,552), and Ogre [27] (19,985/39,856); the mocap video denoted as Mocap [21] (2,494/4,339); and finally the real face videos denoted as Face [26] (28,332/56,516), Face1 [28] (196,446/391,642) and Face2 [28] (196,446/391,614). In addition, we also consider the blending shapes (5,792/10,221) provided by [28]. For quantitative evaluation, we provide the standard 3D error defined by $\bar{e}_{3\text{D}} = \frac{1}{N}\sum_{i=1}^{N} \frac{\|\mathbf{S}_{STS}(\alpha)^i - \mathbf{S}_S(\alpha)^i\|_F}{\|\mathbf{S}_S(\alpha)^i\|_F}$ where $\|.\|_F$ denotes the Frobenius norm. $\mathbf{S}_{STS}(\alpha)^i$ is the retransferred 3D expression over vertex i and $\mathbf{S}_S(\alpha)^i$ is the initial expression value over the same vertex. $\bar{e}_{3\text{D}}$ is computed over 3D shapes that have been previously aligned using Procrustes analysis.

Dataset	Surprise	Kiss	Pain	Angry	Sing	Smile	Sad	$\bar{e}_{3\text{D}}$
Seq3 [26]	7.81	2.50	6.32	1.21	3.82	3.28	1.76	3.81
Mocap [21]	6.30	2.53	3.79	1.43	2.80	2.46	1.42	2.96
Ogre [27]	6.55	2.93	3.82	1.20	5.30	2.46	1.27	3.36
Face [26]	7.01	4.56	5.32	4.42	4.92	5.03	4.40	5.09
$\bar{e}_{3\text{D}}$	6.92	3.13	4.81	2.07	4.21	3.31	2.21	3.81

TABLE I: **Quantitative evaluation of synthetic and real datasets.** Percentage of 3D error over seven types of expressions on four datasets. Two types of analysis: average error per expression and per dataset.

We first evaluate our approach on synthetic face transfer, where the source 3D expressions come from a synthetic dataset [28] (such as surprise, kiss, pain, angry, sing, smile and sad). We then transfer these facial expressions to different 3D models at rest, including the datasets Seq3, Mocap, Ogre, and Face. Figure 2 shows how our algorithm achieves the most challenging of them. It should be noted as our approach even can transfer the facial expression when a large change in topology appears, such as the single-eyed face of a cyclops.

Considering we have no access to a ground truth to measure directly the standard 3D error, we have no other choice but to transfer our expression twice. First, from source to target rest, and second from transferred expression to source rest. Is then when the standard 3D error can be properly computed using the initial expression of the source model as ground truth. The major inconvenience using this procedure is that expression transferring is estimated twice, so the error could be bigger. We provide different results per dataset over different expressions to validate our method (see Table I). Our mean 3D error over the different datasets and challenging expression is 3.8%, however, we want to highlight again that all the quantitative results are produced by a double transferring process so errors may be increased.

In Fig. 4, we graphically interpret the previous analysis. In this case, using the Face sequence as our target, and several 3D expressions as our source information, we apply the double transferring to detect which areas produce more inaccuracies. We represent the corresponding error per vertex for seven primitives in the set of 3D expressions. In general terms, the errors are a product of detail inconsistency on high detailed areas, for instance, a winkle was transferred on the result but it is not on the original 3D.

We also apply our algorithm for on-line facial expression transferring. To do this, we consider the very dense real videos Face1 and Face2. In the first experiment, we compute a rest shape from the Face1 video and consider this model as our 3D target face. Then, we consider the Face2 video as our source information and transfer every frame to the 3D target. A qualitative evaluation of this experiment is displayed in Fig. 3-left. As it can be seen, our approach accurately transfer all frames in the video, even recovering high-frequency details.

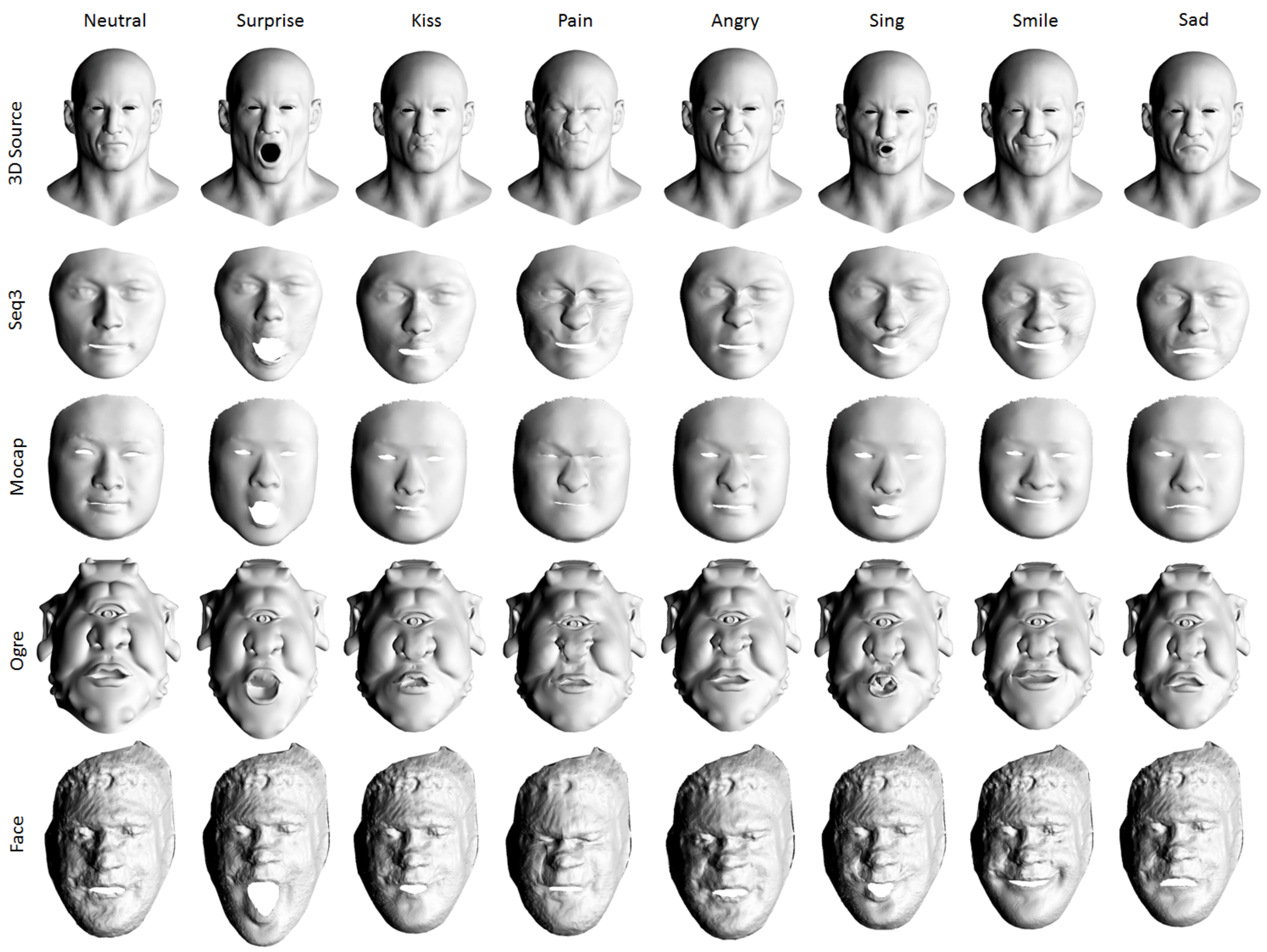

Fig. 2: **Qualitative evaluation of face expression transfer for four different datasets.** In all cases, we display a neutral shape, together with seven expressions denoted surprise, kiss, pain, angry, sing, smile, and sad, respectively. **First row:** A projected view of the 3D source model. **From second to fifth row:** Our 3D estimation in a frontal view of the datasets: Seq3, Mocap, Ogre, and Face, respectively. Observe that our expression transfer algorithm produces very accurate solutions for all cases, even when it is applied for complex shapes like the Ogre dataset. It is worth noting that our approach obtains also nice results for noisy shapes, such as the Face sequence.

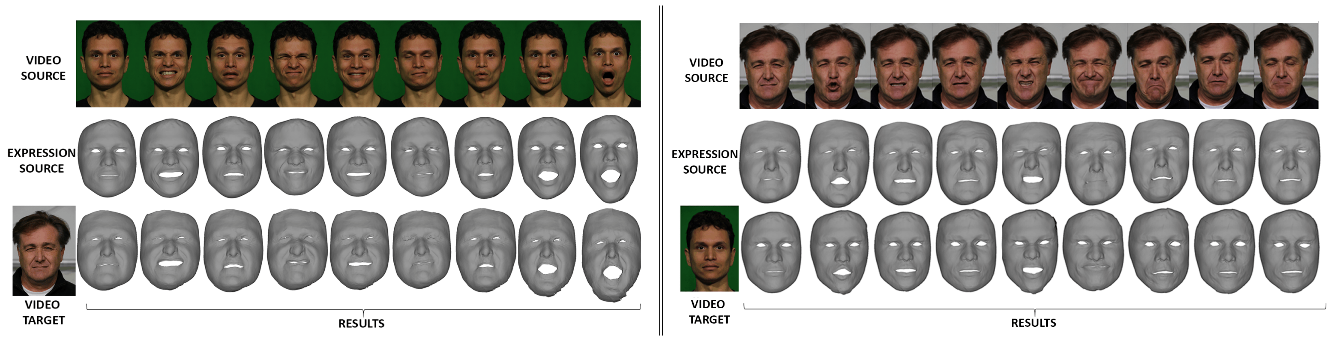

Fig. 3: **Qualitative evaluation of on-line facial expression transfer.** The same information is displayed in both cases. **Top and Middle:** Source image frames and the corresponding 3D estimation. **Bottom:** The left-most picture displays our target face model. We also represent our 3D estimation after the facial expression transfer, considering the source facial gesture in the top row. In all cases, our approach produces high detailed solutions in 3D where original wrinkles and folds are preserved and new expression wrinkles are transferred.

When the datasets Face1 and Face2 are exchanged, i.e., Face1 becomes our source information and Face2 our target, our conclusion is quite similar compared to the previous case, showing the generality of our approach. These results can

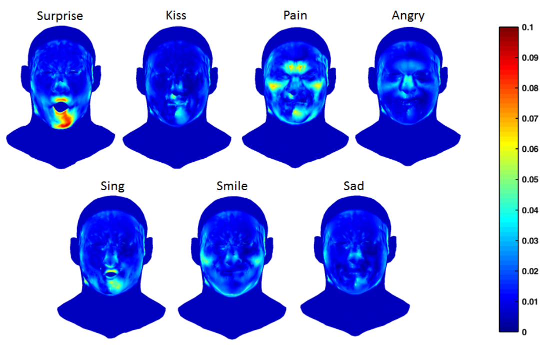

Fig. 4: **Source-target-source transfer.** Once an arbitrary facial expression is transferred from the source to the target, we transfer back from target to source. We display the distribution of the 3D errors for seven facial primitives, considering the Face [26] sequence to define the target model. Bluish areas mean the double transfer is more accurate.

Dataset	Total (s)	Param. (%)	Trans. (%)	Smooth (%)
Seq3 [26]	958.21	90.37	0.23	9.39
Mocap [21]	654.97	95.31	0.04	4.65
Ogre [27]	720.19	80.37	0.22	19.40
Face [26]	1188.09	82.41	0.19	17.40

TABLE II: **Time computation budget.** For every dataset, we provide the total computation time (Total) in seconds to transfer just one expression. Additionally, we also indicate the corresponding partial one for the stages of mapping parametrization (Param), transferring (Trans), and smoothing (Smooth), as a percentage of the total time.

be seen in Fig. 3-right. Finally, Table II reports our time computation budget on a standard desktop computer Intel(R) Xenon(R) CPU ES-1620 V3 at 3.506 GHz, showing the scalability of our approach.

VIII. Conclusion

We have addressed the problem of automatically 2D-to-3D facial expression transfer. To this end, we presented an unsupervised approach that incorporates a sub-region mapping model parametrized with barycentric coordinates. This results in a robust, efficient, and dense technique which can handle topology changes, different mesh resolution, and noisy data. Furthermore, it does not need any training data at all and it has any unreachable expression. We have extensively validated our approach on challenging facial expressions of both synthetic and real videos. We show that it has a superior performance yielding on realistic expressions preserving fine-detail from input face. Also transferring expression wrinkles and folds if the mesh has enough resolution. Our future work is oriented to integrate our approach in data augmentation algorithms to generate 3D dynamic models from the perspective of DL.

Acknowledgments: This work is supported in part by the Spanish Ministry of Science and Innovation under projects FireDMMI TIN2014-56919-C3-2-R, BOSSS TIN2017-89723-P, HuMoUR TIN2017-90086-R, and María de Maeztu Seal of Excellence MDM-2016-0656.

References

[1] Y. Choi, M. Choi, M. Kim, J. Ha, S. Kim, and J. Choo, “Stargan: Unified generative adversarial networks for multi-domain image-to-image translation,” in *arxiv preprint arXiv:1711.09020*, 2017.
[2] H. Averbuch-Elor, D. Cohen-Or, J. Kopf, and M. F. Cohen, “Bringing portraits to life,” *TOG*, vol. 36, no. 4, 2017.
[3] F. Pighin, R. Szeliski, and D. H. Salesin, “Resynthesizing facial animation through 3d model-based tracking,” in *ICCV*, 1999.
[4] P. Garrido, L. Valgaerts, O. Rehmsen, T. Thormahlen, P. Perez, and C. Theobalt, “Automatic face reenactment,” in *CVPR*, 2014.
[5] F. Shi, H.-T. Wu, X. Tong, and J. Chai, “Automatic acquisition of high-fidelity facial performances using monocular videos,” *TOG*, vol. 33, no. 6, p. 222, 2014.
[6] B. Bickel, M. Botsch, R. Angst, W. Matusik, M. Otaduy, H. Pfister, and M. Gross, “Multi-scale capture of facial geometry and motion,” in *TOG*, vol. 26, no. 3, 2007, p. 33.
[7] T. Weise, S. Bouaziz, H. Li, and M. Pauly, “Realtime performance-based facial animation,” in *TOG*, vol. 30, no. 4, 2011, p. 77.
[8] C. Cao, Y. Weng, S. Lin, and K. Zhou, “3d shape regression for real-time facial animation,” *TOG*, vol. 32, no. 4, p. 41, 2013.
[9] D. Vlasic, M. Brand, H. Pfister, and J. Popović, “Face transfer with multilinear models,” in *TOG*, vol. 24, no. 3, 2005, pp. 426–433.
[10] C. Cao, Q. Hou, and K. Zhou, “Displaced dynamic expression regression for real-time facial tracking and animation,” *TOG*, vol. 33, no. 4, p. 43, 2014.
[11] T. Beeler, B. Bickel, P. Beardsley, B. Sumner, and M. Gross, “High-quality single-shot capture of facial geometry,” in *TOG*, vol. 29, no. 4, 2010, p. 40.
[12] D. Bradley, W. Heidrich, T. Popa, and A. Sheffer, “High resolution passive facial performance capture,” in *TOG*, vol. 29, no. 4, 2010, p. 41.
[13] S. Bouaziz, Y. Wang, and M. Pauly, “Online modeling for realtime facial animation,” *TOG*, vol. 32, no. 4, p. 40, 2013.
[14] J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt, “Real-time expression transfer for facial reenactment.” *TOG*, vol. 34, no. 6, pp. 183–1, 2015.
[15] H. Averbuch-Elor, D. Cohen-Or, J. Kopf, and M. F. Cohen, “Bringing portraits to life,” *TOG*, vol. 36, no. 6, p. 196, 2017.
[16] R. W. Sumner and J. Popović, “Deformation transfer for triangle meshes,” *TOG*, vol. 23, no. 3, pp. 399–405, 2004.
[17] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, “Stargan: Unified generative adversarial networks for multi-domain image-to-image translation,” *arXiv preprint arXiv:1711.09020*, 2017.
[18] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, “Face2face: Real-time face capture and reenactment of rgb videos,” in *CVPR*, 2016.
[19] A. Agudo and F. Moreno-Noguer, “Learning shape, motion and elastic models in force space,” in *ICCV*, 2015.
[20] A. Agudo, J. M. M. Montiel, L. Agapito, and B. Calvo, “Modal space: A physics-based model for sequential estimation of time-varying shape from monocular video,” *JMIV*, vol. 57, no. 1, pp. 75–98, 2017.
[21] A. Agudo and F. Moreno-Noguer, “Global model with local interpretation for dynamic shape reconstruction,” in *WACV*, 2017.
[22] C. Tomasi and T. Kanade, “Shape and motion from image streams under orthography: a factorization method,” *IJCV*, vol. 9, no. 2, pp. 137–154, 1992.
[23] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, “Openface: A general-purpose face recognition library with mobile applications,” CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.
[24] R. Garg, A. Roussos, and L. Agapito, “A variational approach to video registration with subspace constraints,” *IJCV*, vol. 104, no. 3, pp. 286–314, 2013.
[25] L. P. Chew, “Constrained delaunay triangulations,” *Algorithmica*, vol. 4, no. 1-4, pp. 97–108, 1989.
[26] R. Garg, A. Roussos, and L. Agapito, “Dense variational reconstruction of non-rigid surfaces from monocular video,” in *CVPR*, 2013.
[27] A. Agudo and F. Moreno-Noguer, “A scalable, efficient, and accurate solution to non-rigid structure from motion,” *CVIU*, vol. 167, no. 2, pp. 121–133, 2018.
[28] P. Garrido, L. Valgaerts, C. Wu, and C. Theobalt, “Reconstructing detailed dynamic face geometry from monocular video.” *TOG*, vol. 32, no. 6, pp. 158–1, 2013.