FaceWarehouse: a 3D Facial Expression Database for Visual Computing

Chen Cao* Yanlin Weng* Shun Zhou* Yiying Tong† Kun Zhou*

*State Key Lab of CAD&CG, Zhejiang University †Michigan State University

***Abstract*—We present FaceWarehouse, a database of 3D facial expressions for visual computing applications. We use Kinect, an off-the-shelf RGBD camera, to capture 150 individuals aged 7–80 from various ethnic backgrounds. For each person, we captured the RGBD data of her different expressions, including the neutral expression and 19 other expressions such as mouth-opening, smile, kiss, etc. For every RGBD raw data record, a set of facial feature points on the color image such as eye corners, mouth contour and the nose tip are automatically localized, and manually adjusted if better accuracy is required. We then deform a template facial mesh to fit the depth data as closely as possible while matching the feature points on the color image to their corresponding points on the mesh. Starting from these fitted face meshes, we construct a set of individual-specific expression blendshapes for each person. These meshes with consistent topology are assembled as a rank-three tensor to build a bilinear face model with two attributes, identity and expression. Compared with previous 3D facial databases, for every person in our database, there is a much richer matching collection of expressions, enabling depiction of most human facial actions. We demonstrate the potential of FaceWarehouse for visual computing with four applications: facial image manipulation, face component transfer, real-time performance-based facial image animation, and facial animation retargeting from video to image.**

***Index Terms*—face modeling, facial animation, face database, mesh deformation, RGBD camera**

I. Introduction

Face models are of great interest to many researchers in computer vision and computer graphics. Different face models are widely used in many applications, including face replacement, face component transfer, image manipulation, face recognition, facial expression recognition and expression analysis. In recent years, 3D face models became popular in increasingly complex visual computing applications due to the 3D nature of human faces, crucial in solutions to problems caused by ambiguities and occlusions. For instance, the latest approaches to face component transfer [1], video face replacement [2], single-view hair modeling [3] are all based on 3D face models.

There exist numerous excellent 3D face databases for various purposes. Blanz and Vetter's 3D morphable model [4] built on an example set of 200 3D face models describing shapes and textures. They then derived a morphable face model, applicable in 3D face reconstruction from a single image. Vlasic et al. [5] presented a multilinear model of 3D face meshes, which contains two separate face models: a bilinear model containing 15 subjects with the same ten facial expressions; a trilinear one containing 16 subjects with five visemes in five different expressions. The multilinear face model can be linked to a face-tracking algorithm to extract pose, expression and viseme parameters from monocular video or film footage, and drive a detailed 3D textured face mesh for a different target identity. Yin et al. [6] developed a 3D facial expression database, which includes both prototypical 3D facial expression shapes and 2D facial textures of 100 subjects with seven universal expressions (i.e., neutral, happiness, surprise, fear, sadness, disgust and anger). The expression database can be used in facial expression recognition and analysis. All of the above face databases contain faces with different identities and expressions, but their expression spaces are not diverse enough for many applications in visual computing, such as the real-time performance-based facial image animation shown in this paper.

We introduce FaceWarehouse, a database of 3D facial expression models for visual computing. With an off-the-shelf RGBD depth camera, raw datasets from 150 individuals aged 7–80 from several ethnicities were captured. For each person, we captured her neutral expression and 19 other expressions, such as mouth-open, smile, angry, kiss, and eye-shut. For each RGBD raw data record, a set of facial feature points on the color image such as eye corners, mouth boundary and nose tip are automatically localized, and manually adjusted if the automatic detection is inaccurate. We then deform a template facial mesh to the depth data as closely as possible while matching the feature points on the color image to their corresponding locations on the mesh. Starting from the 20 fitted meshes (one neutral expression and 19 different expressions) of each person, the individual-specific expression blendshapes of the person are constructed. This blendshape model contains 46 action units as described by Ekman's Facial Action Coding System (FACS) [7], which mimics the combined activation effects of facial muscle groups. It provides a good compromise between realism and control, and adequately describes most expressions of the human face. Owing to the consistent topology of the meshes in the data, we can subsequently organize all the blendshapes of different persons as a rank-three tensor, and construct a bilinear face model with two attributes, identity and expression, through a tensor version of singular value decomposition (SVD).

FaceWarehouse can be used in a wide range of applications in visual computing. In particular, the constructed bilinear face model is used to estimate the identity and expression parameters for faces in images and videos, based on which four applications were developed in this paper. The first application, facial image manipulation, allows users to change geometric facial attributes, such as the size of mouth, the length of face and ethnicity in a single face image. The second application is face component transfer. Given two images of the same person with different expressions, we can transfer local components such as the mouth or eyes from one image to the other, keeping the transferred components compatible with the overall face shape and other components to give the synthetic image a natural look. The third is real-time performance-based facial image animation, allowing a user to animate a face image of a different person by performing in front of an RGBD camera, all in real time. The final application is facial animation retargeting from video to image. Given video footage with a continuously changing face and a still face image as input, we transfer the head motion and facial expression in the video to the still face in the image.

The main contribution of this paper is an extensive face expression database, which contains 150 persons with 47 different facial expressions for each person. To the best of our knowledge, FaceWarehouse is the most comprehensive 3D facial expression database for visual computing to date, providing data sorely needed in a multitude of applications in both computer graphics and computer vision. We will make it publicly available upon the publication of this paper. In addition, we describe how to use the constructed bilinear face model for face identity and expression estimation in facial images and videos. The estimations are accurate enough to support a wide range of applications including animating still face images using real-time RGBD data as well as video footage. We can generate visually plausible facial animations for any portrait image, including those shown in previous work.

In the rest of the paper, we first review some related work in the areas of face model database acquisition and face manipulation applications in Section II. In Section III, we elaborate on the pipeline for the construction of FaceWarehouse. Section IV presents several applications showcasing the potential of FaceWarehouse.

II. Related Work

In this section, we first discuss related work on 2D and 3D face model databases. Some of them focused on neutral expression models, while others contain multiple expressions for dynamical applications. We then discuss applications involving face and head modeling, including face transfer, reanimation, performance tracking in images and video.

Face model databases. As face databases are of great value in face-related research areas in both modeling and validation of methods, many researchers built their own face database for specific applications. Some of these face databases are composed only of 2D face images, while others contain 3D shape information. Some of them contain only one static expression (the neutral face), which is mostly used in applications involving only static faces, e.g. face recognition; while others also contain other expressions, which can be used in face motion-based applications, e.g., face expression recognition and tracking, and face reanimation in still images and video sequences. In the following, we only review several representative existing works, and refer readers to the comprehensive surveys by Gross [8] and Ying et al. [6].

In computer vision, 2D face databases have been widely used as training and/or validation data in face detection and recognition and facial expression analyses. Yin et al.. [6] mentioned that although some systems have been successful, performance degradation remains when handling expressions with large head pose rotation, occlusion and lighting variations. To address the issue, they created a 3D facial expression database. They used a 3D face imaging system to acquire the 3D face surface geometry and surface texture of each subject with seven emotion-related universal expressions. Such a database with few expressions for each person works fine for their face expression recognition and analysis, but may fall short of the diversity required in some applications of visual computing, such as those shown in this paper.

Blanz and Vetter [4] model variations in facial attributes using dense surface geometry and color data. They built the *morphable face model* to describe the procedure of constructing a surface that represents the 3D face from a single image. Furthermore, they supplied a set of controls for intuitive manipulation of appearance attributes (thin/fat, feminine/masculine).

Vlasic et al.. [5] proposed multilinear models in facial expression tracking and transferring. They estimate a model from a data set of three-dimensional face scans that vary in expression, viseme, and identity. This multilinear model decouples the variation into several modes (e.g., identity, expression, viseme) and encodes them consistently. They estimate the model from two geometric datasets: one with 15 identities each performing the same 10 expressions, and the other with 16 identities, each with 5 visemes in 5 expressions ($16 \times 5 \times 5$). To construct their multilinear models from datasets with missing data, they propose to fill the missing combination (e.g., of identity, viseme and expression) by an expectation-maximization approach.

All of these databases contain face data with different identities, and some even with different expressions. However, they may still be inadequate for facial expression parameterization or *rig*. Due to their excellent performance, facial rigs based on blendshape models are particularly popular in expressing facial behaviors. Ekman's Facial Action Coding System [7] helps decompose facial behavior into 46 basic action units. Li et al. [9] developed a method for generating facial blendshape rigs from a set of example

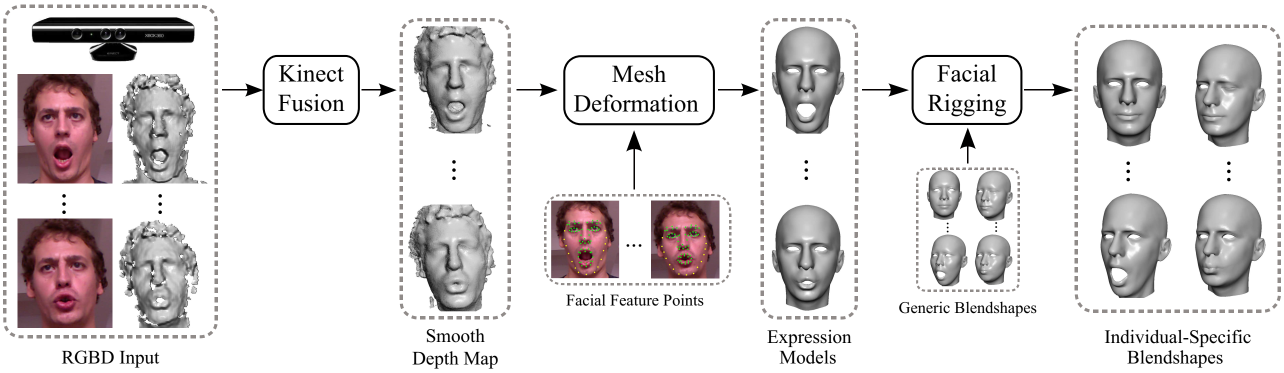


Fig. 1: *The generation process of the individual-specific expression blendshapes for one person.*

poses. Our database contains the full set of 46 expression blendshapes which comprise the linear blendshape model for each person

Face manipulation applications. Face manipulation is a convenient tool for artists and animators to create new facial images or animations from existing materials, and hence of great research interest in computer animation. In the 2D domain, Leyvand et al. [10] enhance the *attractiveness* of human faces in frontal photographs. Given a face image as the target, Bitouk et al. [11] find a most similar face in a large 2D face database and use it to replace the face in the target image to conceal the identity of the face. Joshi et al. [12] improve the quality of personal photos by using a person's favorite photographs as examples.

Recently, 3D face models have become increasingly popular in complex face manipulation. Blanz et al. [13] reanimate the face in a still image or video by transferring mouth movements and expression based on their 3D morphable model [4] and a common expression representation. They also exchange the face between images across large differences in viewpoint and illumination [14]. Given a photo of person A, Shlizerman et al. [2] seek a photo of person B with similar pose and expression from a large database of images on the Internet. Yang et al. [1] derive an expression flow and alignment flow from the 3D morphable model between source and target photos, capable of transferring face components between the images naturally and seamlessly. Shilizerman et al. [15] generate face animations from large image collections. Dale et al. [16] replace the face in a portrait video sequence through a 3D multilinear model [5]. We demonstrate that FaceWarehouse can provide a rich collection of expression data to facilitate and improve these applications.

III. FaceWarehouse

In this section, we describe our pipeline for constructing FaceWarehouse and the techniques involved. We use Microsoft's Kinect System to capture the geometry and texture information of various expressions of each subject. We register the frames from different views of the same expression to generate a smooth, low-noise depth map. The depth maps, together with the RGB images, are used to guide the deformation of a template mesh to generate the expression meshes. Once we obtain all the expression meshes of a single subject, we generate her individual-specific expression blendshapes. Fig. 1 shows the entire pipeline of processing one subject. Finally, the expression blendshapes from all subjects constitute our face database. As we have all the face models in a consistent topology, we can build a bilinear face model with two attributes, identity and expression.

A. Data capture

A Kinect system is used as our only capturing device, capable of producing 640×480 2D images and depth maps at 30 frames per second. With the low-cost small-size acquisition device, we can capture a person's face in a non-intrusive way, as the person being captured is not required to wear any physical markers or be staged in a controlled environment.

For each person, we capture 20 different expressions: the neutral expression and 19 other specific expressions. These expressions are chosen to be common facial motions that vary widely among different individuals. They contain combinations of the facial muscle group action units in FACS and some asymmetric patterns. Specifically, they are mouth stretch, smile, brow lower, brow raiser, anger, jaw left, jaw right, jaw forward, mouth left, mouth right, dimpler, chin raiser, lip puckerer, lip funneler, sadness, lip roll, grin, cheek blowing and eyes closed. We have a guide face mesh for each specific expression, G_0 for the neutral expression, and $G_1, G_2, \ldots, G_{19}$ for the other expressions. The guide models are shown to each subject sequentially, and the person is asked to imitate each expression and rotate her head within a small angle range while keeping the expression fixed, assisted by our staff when necessary.

The advantages of the chosen Kinect system, such as low-cost and mobility, come at the price of low quality in the captured data plagued by severe noise. To reduce noise, we aggregate multiple scans by using the Kinect Fusion algorithm [17] to register the 3D information of a specific expression of a person from different views (captured in the head rotation sequence), and generate a smooth, low-noise depth map. Kinect Fusion works by fusing all the depth data streamed from the camera into a single global implicit surface model, which is then ray traced to generate a smooth depth map for a chosen frame.

B. Expression mesh and individual-specific blendshape generation

From smooth depth maps and corresponding color images, we generate the associated expression meshes. For each expression data, we first use Active Shape Model (ASM) [18] to locate 74 feature points on the color image, including the face contour, eye corners, brow boundary, mouth boundary, nose contour and tip. The automatically detected locations may not be accurate in all cases, especially for those expressions with relatively large deformation (e.g., mouth-open and smile). We thus require a small amount of user interaction to refine the positions of some feature points–the user interaction is as simple as drag-and-dropping the feature points on the image.

The 74 feature points are divided into two categories: the m_i internal feature points (i.e., features on eyes, brows, nose and mouth, c.f. the green points in Fig. 3) located inside the face region, and the m_c contour feature points (the yellow points in Fig. 3). Given the correspondence between the color image and the depth map, we can easily get the corresponding 3D positions from the depth map for internal feature points. We classify all contour feature points in the image as 2D.

Neutral expression. We first generate the face mesh for the neutral expression by using a two-step approach. Blanz and Vetter's morphable model is automatically fitted to produce an initial matching mesh. Then a mesh deformation algorithm is employed to refine this mesh for better matching between the depth map and the feature points.

Blanz and Vetter's morphable model performs Principal Component Analysis (PCA) on 200 neutral face models. Any face can be approximated as a linear combination of the average face and l leading PCA vectors: $V = \overline{F} + \sum_{i=1}^{l} \alpha_i F_i$, where $\overline{F}$ is the average face, and F_i is the i-th PCA vector. Our goal is to compute the coefficients α_i to get the closest mesh in the PCA space. The energy to be minimized for feature point matching is defined as

$$E_{fea} = \sum_{j=1}^{m_i} \left\| \mathbf{v}_{i_j} - \mathbf{c}_j \right\|^2 + \sum_{k=1}^{m_c} \left\| \mathbf{M}_{proj} \mathbf{v}_{c_k} - \mathbf{s}_k \right\|^2. \qquad (1)$$

The first term corresponds to internal feature matching. c_j is the 3D position of the j-th feature point, while $\mathbf{v}_{i_j}$ is its corresponding vertex on the mesh V. The indices for these internal feature points on the mesh are simply marked on the average face in our implementation. The second term is for contour feature matching. $\mathbf{s}_k$ is a 2D feature point on the color image, $\mathbf{v}_{c_k}$ is its corresponding 3D feature vertex on the mesh V, and $\mathbf{M}_{proj}$ is the projection matrix of the camera. We use the method described in [1] to determine the indices of the contour feature points on the mesh V: We first project the face region of V to the image to get the 2D face mesh. Then we find its convex hull to get the points along the contour of the mesh. Among these points, we find the nearest one for each contour feature on the image, and assign it as the corresponding feature point on the mesh.

The energy term for matching the depth map is defined as

$$E_{pos} = \sum_{j=1}^{n_d} \left\| \mathbf{v}_{d_j} - \mathbf{p}_j \right\|^2, \qquad (2)$$

where $\mathbf{v}_{d_j}$ is a mesh vertex, $\mathbf{p}_j$ is the closest point to $\mathbf{v}_{d_j}$ in the depth map, and n_d is the number of the mesh vertices that have valid correspondences in the depth map. Note that not all mesh vertices are accounted for in this energy term as some vertices are occluded and cannot get valid positions from the depth map.

According to [4], another energy term is necessary to regularize the PCA coefficients α_i, based on the estimated probability distribution of a shape defined by α_i,

$$p(\alpha) \sim exp[-\frac{1}{2} \sum (\alpha_i / \sigma_i)^2]. \qquad (3)$$

where σ_i^2 is the eigenvalues of the face covariance matrix from PCA. Let $\Lambda = diag(1/\sigma_1^2, 1/\sigma_2^2, ...1/\sigma_L^2)$, then the Tikhonov regularization energy term is defined as

$$E_{coef} = \frac{1}{2} \alpha^T \Lambda \alpha. \qquad (4)$$

Putting the three energy terms together, the total energy is defined as

$$E_1 = \omega_1 E_{fea} + \omega_2 E_{pos} + \omega_3 E_{coef}, \qquad (5)$$

where ω_1, ω_2 and ω_3 balance the different energy terms. We choose $\omega_1 = 2, \omega_2 = 0.5$ and $\omega_3 = 1$ in our database construction. This energy can be minimized via a sequence of least squares optimizations. The least squares step is iterated 5 to 8 times in our construction. Note that between consecutive iterations, the mesh vertices corresponding to contour feature points in Eq. (1) and each mesh vertex's closest point in Eq. (2) need to be updated.

After an initial mesh is computed, it is refined by a Laplacian-based mesh deformation algorithm [19]. Similar to the optimization process described above, the deformation algorithm tries to minimize E_{fea} and E_{pos} to match the feature points and the depth map. A Laplacian energy is used as the regularization term,

$$E_{lap} = \sum_{i=1}^{n} \left\| L\mathbf{v}_i - \frac{\delta_i}{|L\mathbf{v}_i|} L\mathbf{v}_i \right\|^2, \qquad (6)$$

where L is the discrete Laplacian operator based on the cotangent form introduced in [20], δ_i is the magnitude of the original Laplacian coordinate before deformation, and n is the vertex number of the mesh. The mesh deformation energy is then computed as

$$E_2 = \omega_1' E_{fea} + \omega_2' E_{pos} + \omega_3' E_{lap}. \qquad (7)$$

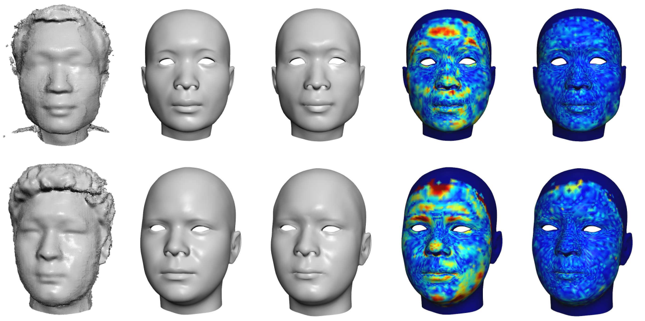


Fig. 2: *Two examples of neutral mesh fitting. From left to right: input depth map; initial mesh produced by the morphable model; refined mesh using mesh deformation; error map of the initial mesh; error map of the refined mesh. The root-mean-square (RMS) errors for the first example are 1.19mm (initial) and 0.39mm (refined), and for the second example: 1.49mm (initial) and 0.41mm (refined).*

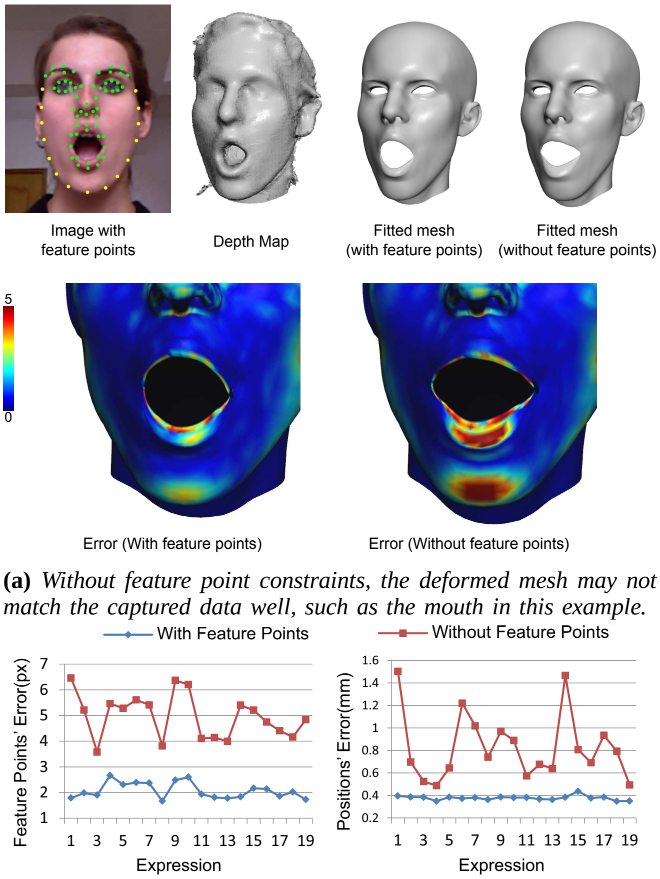
In our construction, we chose $\omega'_1 = 0.5$, $\omega'_2 = \omega'_3 = 1$. This energy can be minimized using the inexact Gauss-Newton method as described in [19].

Mesh deformation helps fine-tune the initial guess. Fig. 2 show two examples of neutral mesh generation. From the figure, we can see that the refined process in the second step drastically reduces the mismatch, resulting in a better matching face mesh.

Other expressions. Once we obtain the face mesh S_0 for the neutral expression, we proceed to compute the face meshes $S_1, S_2, \ldots S_{19}$ for the other 19 expressions. We first use the deformation transfer algorithm described in [21] to generate the initial meshes for these expressions so that the deformation from S_0 to S_i $(i = 1 \ldots 19)$ mimics the deformation from the guide model G_0 to G_i as much as possible. The same mesh deformation algorithm described above is then used to refine these initial meshes.

The mesh deformation algorithm uses all facial feature points on the color images as additional position constraints. We found in our experiments that these constraints not only greatly reduce the matching errors between the image feature points and their corresponding mesh vertices but also help avoid local minima in the deformation process and improve the matching between the mesh and the depth map, as demonstrated in Fig. 3.

Individual-specific expression blendshapes. From the expression meshes generated for each person, we can use the example-based facial rigging algorithm proposed by Li et al. [9] to build a linear blendershape model representing the facial expression space of this person. The result is a neutral face plus 46 FACS blendshapes $\mathbf{B} = \{B_0, B_1, \ldots, B_{46}\}$ for each person, capable of replicating most human expressions through linear interpolation of the blendshapes. In other words, an expression H of the person can be expressed by a linear combination of the blendshapes: $H = B_0 + \sum_{i=1}^{46} \alpha_i (B_i - B_0)$, where α_i is the weight measuring how much the neutral face B_0 is deformed toward B_i. The rigging algorithm begins with a generic blendshape model $\mathbf{A} = \{A_0, A_1, \ldots, A_{46}\}$, and employs an optimization procedure to minimize the difference between each expression mesh S_j and the linear combination of B_i with the known weights for expression j as well as the difference between the relative deformation from B_0 to B_i and that from A_0 to A_i. See [9] for details of the algorithm.



(a) *Without feature point constraints, the deformed mesh may not match the captured data well, such as the mouth in this example.*

(b) *The lack of feature point constraints leads to results trapped in local minima in the deformation process and causes greater matching error, for both the feature points of color image and the depth map.*

Fig. 3: *The effect of feature point constraints in mesh deformation.*

C. Blinear face model

We obtained the facial geometry of 150 persons and each contains the same 47 facial expressions (1 neutral and 46 others) by using the procedure described in the previous section. All these face meshes share the same topology and thus the same number of vertices. Similar to [5], we can assemble the dataset into a rank-three (3-mode) *data tensor* T (11K vertices × 150 identities × 47 expressions). The data tensor is arranged in an obvious fashion, so that each slice with varying second factor and fixed third factor contains face vectors with the same expression (for different identities), and each slice with varying third factor and fixed second factor contains the same identity (with different expressions).

We use the N-mode singular value decomposition (SVD) to decompose the tensor. As most visual applications only need to synthesize the entire face, we perform the decomposition without factoring along the vertex mode (mode-1). The N-mode SVD process is represented as

$$T \times_2 \mathbf{U}_{id}^T \times_3 \mathbf{U}_{exp}^T = C, \tag{8}$$

where T is the data tensor and C is called the *core tensor*. $\mathbf{U}_{id}$ and $\mathbf{U}_{exp}$ are orthonormal transform matrices, which contain the left singular vectors of the $2nd$ mode (identity) space and $3rd$ mode (expression) space respectively. N-mode SVD helps "rotate" the data tensor and sort the variance of C in decreasing order for each mode. This allows us to truncate the insignificant components of C and get a reduced model of the dataset to approximate the original data tensor as

$$T \simeq C_r \times_2 \check{\mathbf{U}}_{id} \times_3 \check{\mathbf{U}}_{exp}, \tag{9}$$

where C_r is the reduced core tensor produced by keeping the top-left corner of the original core tensor. $\check{\mathbf{U}}_{id}$ and $\check{\mathbf{U}}_{exp}$ are the truncated matrices from $\mathbf{U}_{id}$ and $\mathbf{U}_{exp}$ by removing the trailing columns.

We call C_r the bilinear face model for FaceWarehouse. With C_r, any facial expression of any person can be approximated by the tensor contraction

$$V = C_r \times_2 \mathbf{w}_{id}^T \times_3 \mathbf{w}_{exp}^T, \tag{10}$$

where $\mathbf{w}_{id}$ and $\mathbf{w}_{exp}$ are the column vectors of identity weights and expression weights, respectively.

Fig. 4 shows an example of fitting a face mesh using different numbers of components in the core tensor. We found that choosing 50 knobs for identity and 25 knobs for expression provides satisfactory approximation results in all of our applications.

IV. Applications

FaceWarehouse can be employed in various visual computing applications. In this section, we show four example applications: facial image manipulation, face component transfer, real-time performance-based facial image animation, and facial animation retargeting from video to image. Refer to the supplementary material for the video demo.

A. Facial image manipulation

In this application, the user can manipulate facial attributes, such as the length of face, the size of mouth, the height of nose and ethnicity, directly in the single input face image. As we only have the two attributes of identity and expression in FaceWarehouse, we first learn a linear regression model that maps a set of user-specified facial attributes to the identity attribute in the bilinear face model. We then compute the identity and expression weights in our bilinear face model for the input face image. The changes to the user-defined attributes are mapped to the identity weights via the linear regression model, and then a new 3D face mesh is reconstructed based on these weights. This new 3D face mesh is finally rendered with color textures from the input image to generate a new face image with the relevant features changed.

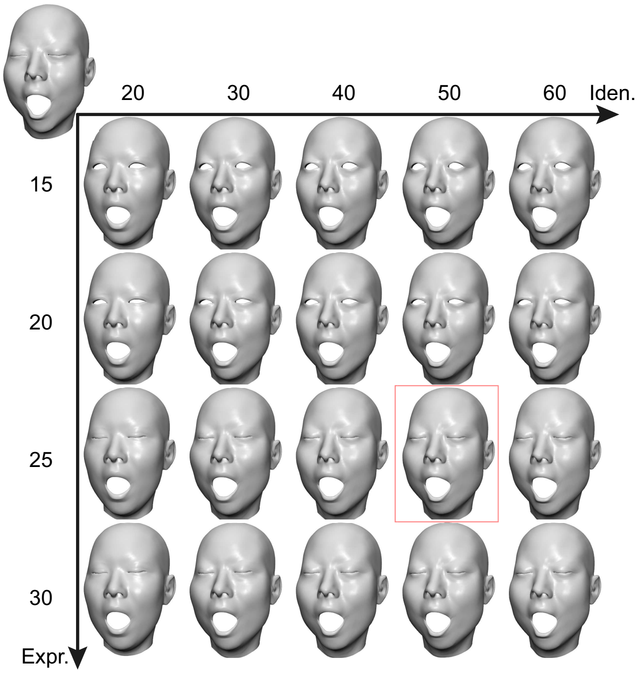


Fig. 4: *Fitting a facial expression mesh with our bilinear model with different numbers of components. Top left is the input mesh, and the following shows the fitting results using different numbers of components in identity attribute and expression attribute.*

Facial feature analysis. To analyze facial attributes used in natrual language (e.g., the width of mouth, the length of face), we use the algorithm of multi-variate linear regression [22] to map these attributes to the identity attribute in our bilinear face model. For every person captured in FaceWarehouse, we have his (or her) identity weights (a k-D vector $\mathbf{w}_{id}$) and l user-specified attributes $\{f_1, f_2, ..., f_l\}$ which are calculated from the face geometry of this person. We need to construct a $k \times (l+1)$ matrix $\mathbf{M}_{fea}$ mapping these user-specified attributes to the identity weights,

$$\mathbf{M}_{fea}[f_1, ..., f_l, 1]^T = \mathbf{w}_{id}. \tag{11}$$

We assemble the vectors of identity weights and the vectors of user-defined attributes of all 150 persons into two matrices, i.e., $\mathbf{W}_{id}$ ($k \times 150$) and $\mathbf{F}$ ($(l+1) \times 150$) respectively. The mapping matrix $\mathbf{M}_{fea}$ can be solved as

$$\mathbf{M}_{fea} = \mathbf{W}_{id}\mathbf{F}^+ \tag{12}$$

where $\mathbf{F}^+$ is the left pseudoinverse of $\mathbf{F}$, i.e., $\mathbf{F}^+ = \mathbf{F}^T(\mathbf{F}\mathbf{F}^T)^{-1}$.

With $\mathbf{M}_{fea}$, we can directly map the changes of the user-specified attributes $\Delta\mathbf{f}$ to the changes of identity weights $\Delta\mathbf{w}_{id}$, i.e., $\Delta\mathbf{w}_{id} = \mathbf{M}_{fea}\Delta\mathbf{f}$. By adding $\Delta\mathbf{w}_{id}$ to the identity

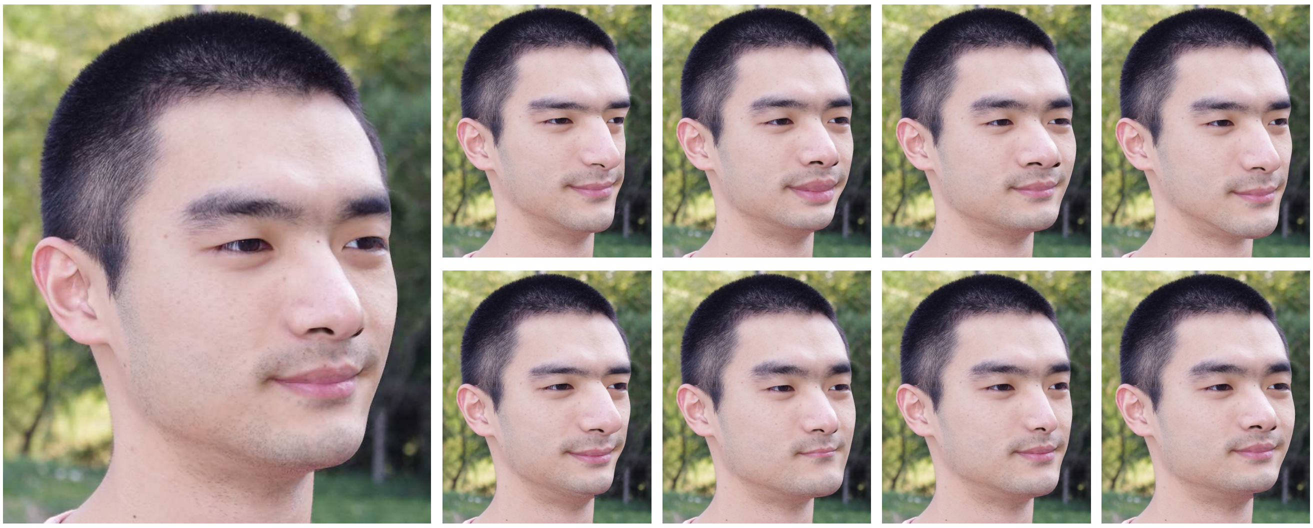


Fig. 5: *Facial image manipulation. We can edit the facial attributes in the still image. Left: the original image. Top row, from left to right: more Caucasian-like, bigger mouth, wider nose, longer face. Bottom row, from left to right: higher nose, smaller mouth, narrower nose, shorter face.*

weights of the input face image, we can generate a new face with related attributes changed appropriately.

Fitting 3D face mesh to image. To calculate a 3D face mesh using our bilinear model that can match the face image well, we first localize a set of facial feature points in the image in the same way as we suggest in Section III-B. Then we estimate the rigid transformation of the face model as well as the identity and expression weights in the bilinear face model to minimize the matching error between the feature points on the image and the face mesh.

Following previous work [5], we assume that the camera projection is weakly perspective. Every mesh vertex $\mathbf{v}_k$ is projected to the image space as

$$\mathbf{p}_k = s\mathbf{R}\mathbf{v}_k + \mathbf{t} \tag{13}$$

where the rigid transformation consists of a scaling factor s, a 3D rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$. The mesh vertex position $\mathbf{v}_k$ can be computed from the bilinear face model according to Eq. (10).

The matching error of the feature points on the image and the mesh is defined as

$$E_k = \frac{1}{2} \cdot \|s\mathbf{R}(C_r \times_2 \mathbf{w}_{id}^T \times_3 \mathbf{w}_{exp}^T)^{(k)} + \mathbf{t} - \mathbf{s}^{(k)}\|^2, \tag{14}$$

where $\mathbf{s}^{(k)}$ is the feature point positions on the image.

This energy can be easily minimized using the coordinate-descent method as described in [5].

B. Face component transfer

This application performs face component copy-and-paste to modify the expression in a facial image. It takes two images of the same person as input: one is the target photo that contains an undesirable expression and the other one is the reference image that contains the desired expression, such as smiling. As modifying expression causes global changes in one's face, if we directly copy the local component from the reference image and paste/blend it to the target, the transferred component may not be compatible with the face contour or other components in the target image. A better approach proposed by Yang et al. [1] is to use 3D face models to guide the component transfer process. We follow this approach and use our bilinear face model to synthesize 3D face models matching the input images.

As the two input images represent the same person, their identity weights $\mathbf{w}_{id}^T$ should be the same. We therefore need to compute the unified identity weights $\mathbf{w}_{id}^T$ and the expression weights ($\mathbf{w}_{exp-1}^T$ and $\mathbf{w}_{exp-2}^T$) for the two images. This can be done by minimizing the following energy

$$E_k^{joint} = \frac{1}{2}\sum_{j=1}^{2} \|s_j\mathbf{R}_j(C_r \times_2 \mathbf{w}_{id}^T \times_3 \mathbf{w}_{exp-j}^T)^{(k)} + \mathbf{t}_j - \mathbf{s}_j^{(k)}\|^2. \tag{15}$$

We first use the method described in the last section to compute an initial estimation of the identity and expression weights for each image separately. Then we fix $\mathbf{w}_{exp-1}$ and $\mathbf{w}_{exp-2}$ and compute the unified identity weights $\mathbf{w}_{id}^T$ by minimizing Eq. (15). Next $\mathbf{w}_{exp-1}$ and $\mathbf{w}_{exp-2}$ are solved again separately with $\mathbf{w}_{id}^T$ fixed. The latter two steps are performed iteratively until the fitting results converge. In our experiments, three iterations produce satisfactory results.

The two fitted face meshes can be reconstructed as

$$\begin{aligned} V_1 &= C_r \times_2 \mathbf{w}_{id}^T \times_3 \mathbf{w}_{exp-1}^T, \\ V_2 &= C_r \times_2 \mathbf{w}_{id}^T \times_3 \mathbf{w}_{exp-2}^T. \end{aligned} \tag{16}$$

Following [1], we use the two face meshes to calculate a 2D expression flow in the target image, which warps

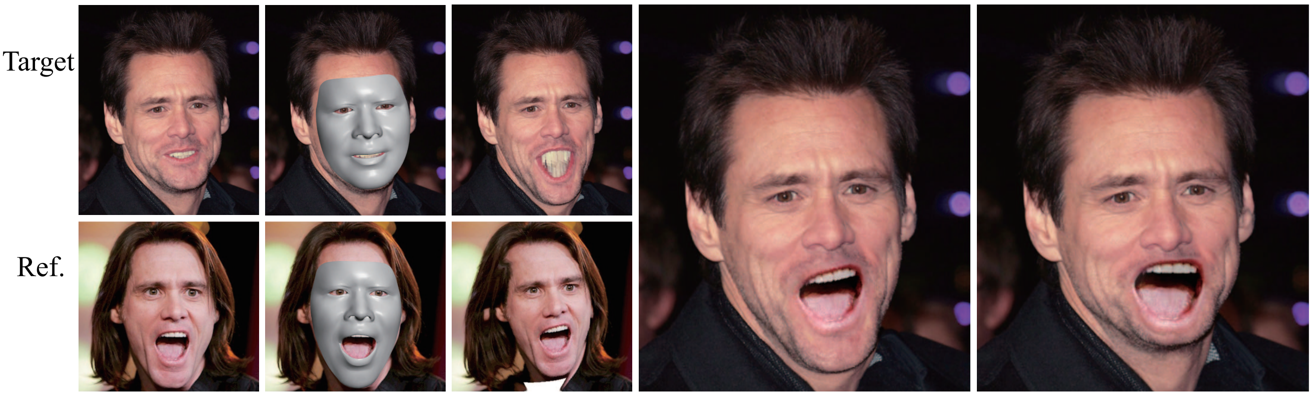
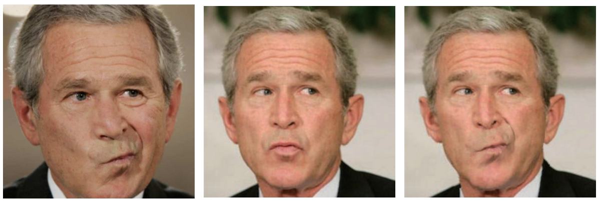


Fig. 6: *Face component transfer. First column: two input images of the same person with different expressions. Second column: fitted meshes. Third column: target image warped by the expression flow and the reference image warped by the alignment flow. Fourth column: the transferred result by our method. Last column: the result produced by the 2D copy-and-paste method.*



(a) *Reference* **(b)** *Target* **(c)** *Transfer*

Fig. 7: *An example of face component transfer with an asymmetric expression.*

the target face to match the desired expression. A 2D alignment flow is also calculated from these two meshes to warp the reference face to an appropriate size and position for transferring. Finally we select a crop region from the warped reference image and blend it to the warped target image to generate the transferred result. Fig. 6 shows one mouth-open example; our method produces a much more realistic result than the 2D copy-and-paste method. Fig. 7 shows another example with an asymmetric expression.

C. Real-time performance-based facial image animation

In this application, a still face image is animated in real time by the facial performance of an arbitrary user (see Fig. 8). Again, we first use the algorithm described in Section IV-A to compute the identity and expression weights to produce a 3D face mesh matching the feature points in the input image. We then generate the individual-specific expression blendshapes for this face mesh using the bilinear face model. Finally, we implement the Kinect-based facial animation system to capture the user's facial expressions expressed as blendshape coefficients, which are then transferred to the individual face model fitted for the image.

Assuming that the identity weights computed from the face image are $\mathbf{w}_{id}$, we can construct the expression blendshapes for the person of identity $\mathbf{w}_{id}$ as follows

$$B_i = C_r \times_2 \mathbf{w}_{id} \times_3 (\check{\mathbf{U}}_{exp}\mathbf{d}_i), \quad 0 \le i \le 47, \qquad (17)$$

where $\check{\mathbf{U}}_{exp}$ is the truncated transform matrix for the expression mode as described in Section III-C, and $\mathbf{d}_i$ is the expression weight vector with value 1 for the i-th element and 0 for other elements.

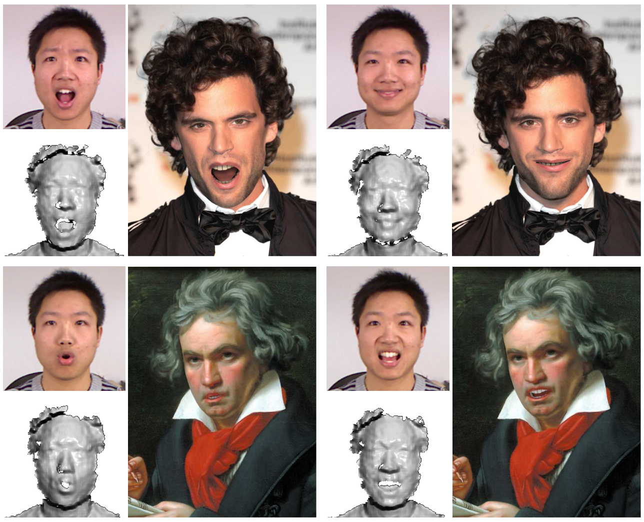


Fig. 8: *Using a Kinect camera to track a user's facial expressions, which are then transferred to a still facial image, all in real time. This application allows a user to create an image avatar from a single facial image of another person.*

The generated expression blendshapes, which we call *image blendshapes*, can then be used to generate new expressions of the same identity by setting different coefficients β_i for these blendshapes: $V = B_0 + \sum_{i=1}^{46} \beta_i (B_i - B_0)$.

We then use a real-time performance-based facial animation system [23] to capture the dynamic expressions of an arbitrary user, who has another set of expression blendshapes ($\mathbf{U} = \{U_0, U_1, ..., U_{46}\}$) constructed by the system

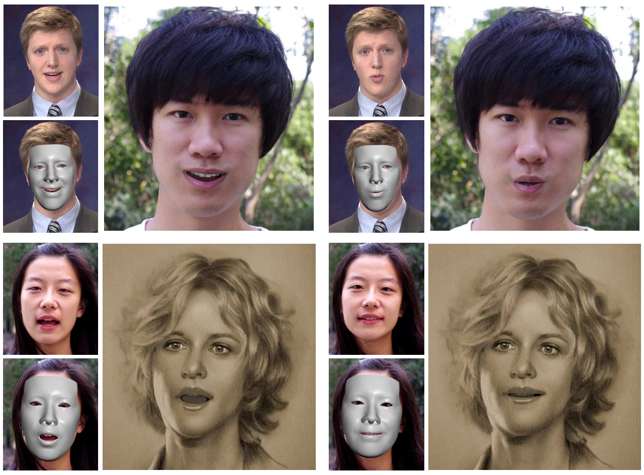


Fig. 9: *Facial animation retargeting from video to image.*

during preprocessing. The system is able to track the rigid transformation of the user's head and the facial expressions expressed in the format of blendshapes coefficients β_i, which are then easily transferred to image blendshapes **B** to synthesize facial animations that mimic the user's performance.

To render the image animations in a realistic manner, the hair and teeth need to be processed in a proper way. We use a single-view hair modeling technique [3] to reconstruct a strand-based 3D hair model, which is then transformed together with the face mesh and rendered into the image. The teeth are handled using the algorithm described in [23]. Starting from a generic teeth model, we deform it to match the 3D face model. During animation, the motion of the teeth is easy to simulate: the upper jaw teeth are connected and moved with the upper part of the face while the lower jaw teeth are connected and moved with the tip of the chin.

D. Facial animation retargeting from video to image

The final application takes a video clip containing a continuously changing face as input, extracts the coefficients of expression blendshapes in all frames and retargets them to a still face image (see Fig. 9). It needs to estimate the face identity and construct the expression blendshapes for both the face video and face image. The expression coefficients fitted for the face video are then transferred to the face image.

The face identity and expression of the image can be estimated using the algorithm described in Section IV-A. For the video, we need to fit a unified face identity for all frames using a simple extension of the joint fitting algorithm described in Section IV-B. We first locate the facial feature points on all frames, and then extend the joint fitting algorithm to multiple frames to account for the matching errors in all frames. After the face identities of the video and image are fixed, we use the method described in Section IV-C to construct their expression blendshapes.

V. Conclusion and Future Work

We introduce FaceWarehouse, a 3D facial expression database for visual computing applications. The database contains the facial geometry and texture of 150 subjects, each with 20 expressions. This raw dataset is used to construct 47 expression blendshapes for each person, capable of representing most expressions of human faces. All these blendshapes are then assembled into a rank-three tensor, which is decomposed to build a bilinear face model. This bilinear face model can be used to accurately estimate face identities and expressions for facial images and videos. We demonstrate its relevance in a wide range of applications, such as facial image manipulation, face component transfer, real-time performance-based facial image animation, and facial animation retargeting from video to image. We expect many other applications to benefit from FaceWarehouse in the future, such as face tracking in motion capture, expression recognition and analysis.

Due to the low precision in depth information provided by the current Kinect system, our face data do not contain detailed facial geometries such as wrinkles. In the future, it is possible to employ capturing techniques for high-quality facial geometry (e.g., [24], [25], [26]) to acquire data with finer details.

References

[1] F. Yang, J. Wang, E. Shechtman, L. Bourdev, and D. Metaxas, "Expression flow for 3d-aware face component transfer," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 60:1–60:10, Jul. 2011.

[2] I. Kemelmacher-Shlizerman, A. Sankar, E. Shechtman, and S. M. Seitz, "Being john malkovich," in *Proceedings of the 11th European conference on Computer vision: Part I*, ser. ECCV'10, 2010, pp. 341–353.

[3] M. Chai, L. Wang, Y. Weng, Y. Yu, B. Guo, and K. Zhou, "Single-view hair modeling for portrait manipulation," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 116:1–116:8, Jul. 2012.

[4] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Proceedings of ACM SIGGRAPH*, 1999, pp. 187–194.

[5] D. Vlasic, M. Brand, H. Pfister, and J. Popović, "Face transfer with multilinear models," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 426–433, Jul. 2005.

[6] L. Yin, X. Wei, Y. Sun, J. Wang, and M. J. Rosato, "A 3d facial expression database for facial behavior research," in *Proc. IEEE Intl Conf. Face and Gesture Recognition*, 2006, pp. 211–216.

[7] P. Ekman and W. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement.* Consulting Psychologists Press, 1978.

[8] R. Gross, "Face databases," in *Handbook of Face Recognition*, A. S.Li, Ed. New York: Springer, February 2005.

[9] H. Li, T. Weise, and M. Pauly, "Example-based facial rigging," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 32:1–32:6, Jul. 2010.

[10] T. Leyvand, D. Cohen-Or, G. Dror, and D. Lischinski, "Data-driven enhancement of facial attractiveness," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 38:1–38:9, Aug. 2008.

[11] D. Bitouk, N. Kumar, S. Dhillon, P. Belhumeur, and S. K. Nayar, "Face swapping: automatically replacing faces in photographs," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 39:1–39:8, Aug. 2008.

[12] N. Joshi, W. Matusik, E. H. Adelson, and D. J. Kriegman, "Personal photo enhancement using example images," *ACM Trans. Graph.*, vol. 29, no. 2, pp. 12:1–12:15, Apr. 2010.

[13] V. Blanz, C. Basso, T. Poggio, and T. Vetter, "Reanimating faces in images and video," *Computer Graphics Forum*, vol. 22, no. 3, pp. 641–650, 2003.

[14] V. Blanz, K. Scherbaum, T. Vetter, and H.-P. Seidel, "Exchanging faces in images," *Computer Graphics Forum*, vol. 23, no. 3, pp. 669–676, 2004.

[15] I. Kemelmacher-Shlizerman, E. Shechtman, R. Garg, and S. M. Seitz, "Exploring photobios," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 61:1–61:10, Jul. 2011.

[16] K. Dale, K. Sunkavalli, M. K. Johnson, D. Vlasic, W. Matusik, and H. Pfister, "Video face replacement," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 130:1–130:10, Dec. 2011.

[17] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ser. ISMAR '11, 2011, pp. 127–136.

[18] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models-their training and application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38 – 59, 1995.

[19] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum, "Subspace gradient domain mesh deformation," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1126–1134, Jul. 2006.

[20] M. Desbrun, M. Meyer, P. Schroder, and A. H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," in *Proceedings of ACM SIGGRAPH*, 1999, pp. 317–324.

[21] R. W. Sumner and J. Popović, "Deformation transfer for triangle meshes," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 399–405, Aug. 2004.

[22] B. Allen, B. Curless, and Z. Popović, "The space of human body shapes: reconstruction and parameterization from range scans," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 587–594, Jul. 2003.

[23] T. Weise, S. Bouaziz, H. Li, and M. Pauly, "Realtime performance-based facial animation," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 77:1–77:10, Jul. 2011.

[24] H. Huang, J. Chai, X. Tong, and H.-T. Wu, "Leveraging motion capture and 3d scanning for high-fidelity facial performance acquisition," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 74:1–74:10, Jul. 2011.

[25] T. Beeler, B. Bickel, P. Beardsley, B. Sumner, and M. Gross, "High-quality single-shot capture of facial geometry," *ACM Trans. Graph.*, vol. 29, pp. 40:1–40:9, July 2010.

[26] 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," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 75:1–75:10, Jul. 2011.

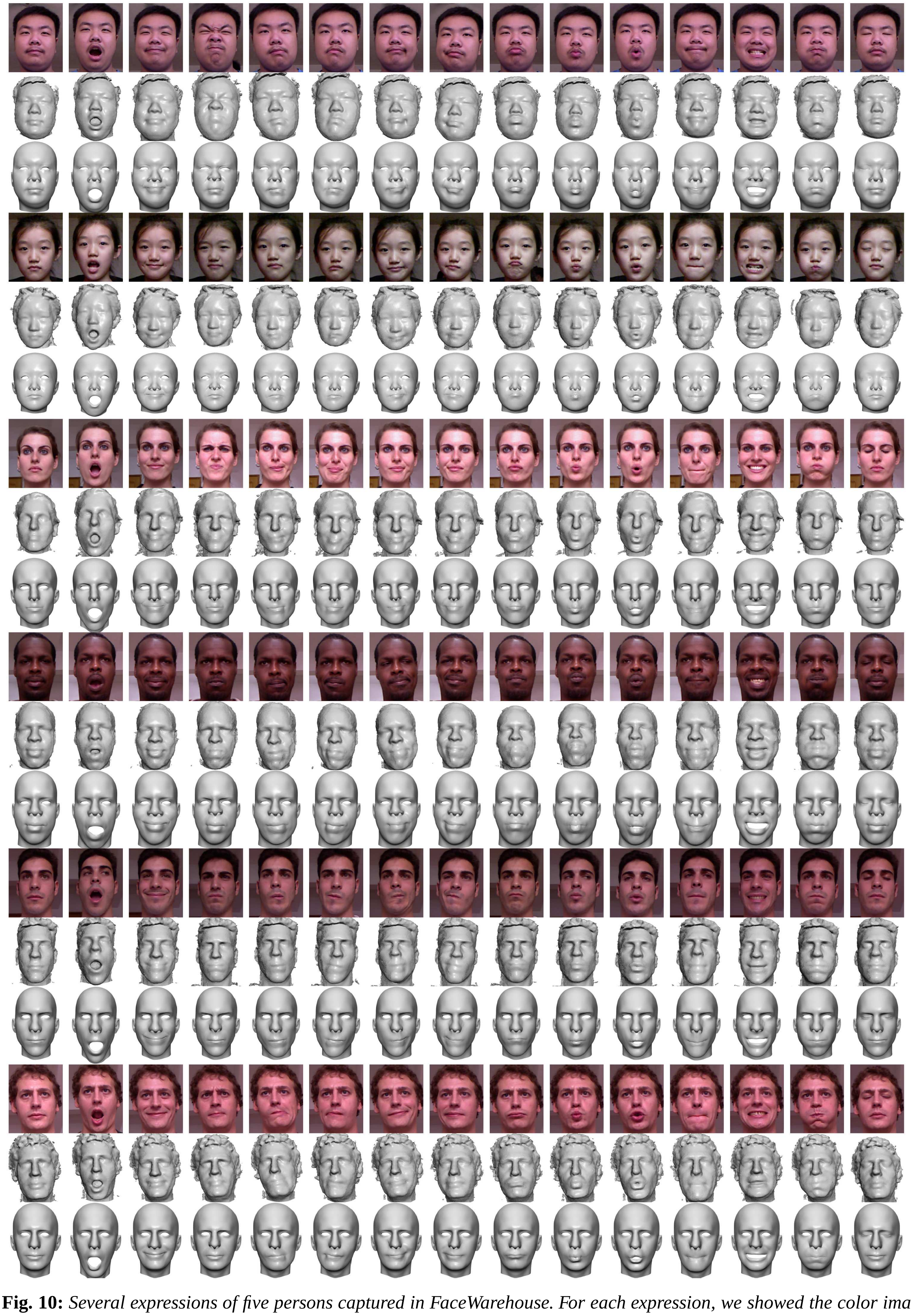


Fig. 10: *Several expressions of five persons captured in FaceWarehouse. For each expression, we showed the color image, the depth map and the reconstructed mesh.*