Self-supervised CNN for Unconstrained 3D Facial Performance Capture from an RGB-D Camera

Yudong Guo, Juyong Zhang†, Lin Cai, Jianfei Cai and Jianmin Zheng

Abstract—We present a novel method for real-time 3D facial performance capture with consumer-level RGB-D sensors. Our capturing system is targeted at robust and stable 3D face capturing in the wild, in which the RGB-D facial data contain noise, imperfection and occlusion, and often exhibit high variability in motion, pose, expression and lighting conditions, thus posing great challenges. The technical contribution is a self-supervised deep learning framework, which is trained directly from raw RGB-D data. The key novelties include: (1) learning both the core tensor and the parameters for refining our parametric face model; (2) using vertex displacement and UV map for learning surface detail; (3) designing the loss function by incorporating temporal coherence and same identity constraints based on pairs of RGB-D images and utilizing sparse norms, in addition to the conventional terms for photo-consistency, feature similarity, regularization as well as geometry consistency; and (4) augmenting the training data set in new ways. The method is demonstrated in a live setup that runs in real-time on a smartphone and an RGB-D sensor. Extensive experiments show that our method is robust to severe occlusion, fast motion, large rotation, exaggerated facial expressions and diverse lighting.

Index Terms—facial performance capture, self-supervised learning, convolutional neural networks

I. INTRODUCTION

This paper considers the problem of real-time, high-quality 3D facial performance capture in unconstrained environment using a consumer-level RGB-D sensor. Note that 3D face capture from RGB videos has been well studied [54], [17], [11], [10], but real-time RGB based methods still have difficulty to achieve high-quality and robust facial performance capture due to limited information from the input RGB images (see Fig. 8). RGB-D sensors, on the other hand, provide not only appearance information but also depth information that is robust to illumination changes and occlusions. Hence RGB-D based capturing has recently received increasing attention, especially with consumer-grade depth cameras becoming popular. For example, some newly released smart phones like Apple’s iPhone X are equipped with an RGB-D camera, which stimulates the call for more efficient and pragmatic RGB-D face capturing solutions on mobile devices. While smart phones require minimal hardware setup, their acquisition is likely in unconstrained environment and their computation power is limited, which pose technical challenges. Therefore the problem of developing real-time, high-quality 3D face capture in unconstrained environment is of great interest and its solution will facilitate various applications such as personalized gaming, make-up apps, social media and telepresence at the consumer-level.

Since Weise et al. [58] presented a seminal work for real-time facial performance capture based on RGB-D camera, several enhancements have been proposed [7], [32], [13], [53]. However, a few limitations exist in these works. First, the tracking and reconstruction is formulated as a non-linear optimization problem and consequently real time performance can only be achieved on PC, but not on mobile devices, as pointed out in [53]. Second, a parametric model defined based on a fixed PCA (Principal Component Analysis) basis is used for either identity [58], [7], [32] or both identity and expression [53]. Such a parametric model may not match the test 3D face well. Thus personalized blendshape is built to better represent the test 3D face, which requires user-specific calibration or manual assistance. Third, since the low rank representations only capture the basic shape of the face, the reconstructed face usually lacks facial detail such as wrinkles.

Our work is inspired by the great success of deep learning technology, particularly convolution neural networks (CNN), which has caused paradigm-shift in many research fields due to its powerfulness in data-driven modelling. Deep learning is very suitable for applications that involve complex optimization but require fast processing speed, since learning based solutions can shift complex optimization to the offline training stage while the online inference stage can be very fast. Recently, deep learning based methods have been proposed for 3D face reconstruction based on RGB inputs. [41] and [20] propose a coarse-to-fine network architecture where a coarse-scale CNN is trained to regress the parameters of a parametric face model and a fine-scale CNN is trained to recover the face detail. [52] proposes an unsupervised CNN structure including a CNN encoder and a fixed rendering module as decoder and trains the structure end-to-end for 3D face reconstruction from RGB input. [51] further improves the reconstruction accuracy by learning the corrective basis from sparsely labeled in-the-wild images.

Despite the great progress, learning based RGB methods still inherit the limitations from traditional optimization based RGB methods. For example, RGB-based methods only have appearance information and they cannot perform well or even fail in dark environment, while RGB-D based methods can still work as shown in Fig. 8. With the help of additional depth information, RGB-D based methods could achieve better and more robust performance, compared to RGB-based methods. Motivated by these observations, we aim to introduce a CNN based solution with RGB-D data.

Yudong Guo, Juyong Zhang and Lin Cai are with School of Mathematical Sciences, University of Science and Technology of China. Jianfei Cai and Jianmin Zheng are with School of Computer Science and Engineering, Nanyang Technological University.

†Corresponding author. Email: juyong@ustc.edu.cn.
It is however non-trivial to either convert existing optimization-based RGB-D methods to learning-based approaches or extend existing deep learning based RGB face capture methods to RGB-D scenarios. For example, [20] requires to synthesize a large number of photo-realistic face images with corresponding 3D geometry information based on inverse rendering in order to train their coarse-scale CNN and fine-scale CNN in a fully supervised manner. Nevertheless, synthesizing another large set of RGB-D photo-realistic face images with high-quality 3D geometry information would be too time-consuming.

In this paper, we propose a CNN framework for real-time RGB-D based 3D face capture. The framework follows the coarse-to-fine network architecture and consists of a medium-scale CNN to regress a medium-scale face model and a fine-scale CNN to recover the surface details. Both CNN models are trained in a completely self-supervised manner in that the face shape and details are automatically learned from large-scale unlabelled RGB-D data. Different from existing methods which use a fixed PCA basis for the parametric model, our method learns both the basis (or core tensor) and the parameter coefficients from the large scale training data set. In this way, the refined parametric face model can better capture the target 3D face shape. Moreover, we extend our framework to the scenario with RGB inputs, where our novel idea is to use both RGB and depth during training while only using RGB during testing for facial performance capture. Specifically, during training we utilize the captured depth data as an additional supervision information at the output. Experiments show that such additional supervision from depth greatly improves 3D face reconstruction performance from RGB inputs.

Apart from the geometry consistency, photo-consistency and regularization, our training process also takes into account other factors including temporal coherence and same identity constraints, the use of sparse norms such as $\ell_{2,1}$ norm and $\ell_1$ norm, and vertex displacement for modeling surface details, which lead to robust and high-quality tracking and reconstruction.

We have demonstrated our method in a live setup where the hardware includes a Huawei Mate 10 smartphone and an RGB-D camera as shown in Fig. 1. The experiments were conducted with different subjects, diverse lighting conditions, challenging head motions, exaggerated expressions and severe occlusions. For all these test examples, our medium-scale reconstruction can run on mobile devices with half-precision float in real time.

---

1 We will release the dataset after clearance of copyright issues.
The quantitative comparison with the state-of-the-art shows that our method performs better.

II. RELATED WORK

Facial performance capture is a fundamental topic in computer graphics and animation. It is well-established in the film industry for animation production. There is rich literature in this field. A nice overview of fundamental techniques for performance-driven facial animation is given in [39], [63]. Traditional film and game production uses controlled studio conditions for high-quality capturing. Facial makers [18], [5] or calibrated camera arrays [8], [3] are also used for producing robust features or high-quality data. Recently many methods that are easily adaptable and suitable for consumer-level applications are developed. This section briefly reviews those techniques that are related to our work.

Facial Performance Capture from RGB Inputs. Monocular RGB images or video can be easily obtained and many methods have been proposed to reconstruct faces from monocular RGB video. A popular way is to fit the input images using a parametric face model such as the 3D Morphable Model (3DMM) [6]. The reconstructed model can be further refined by recovering 3D shape from shading variation [28], [27]. For instance, in [16], a template is deformed to a 3D model created by a binocular stereo approach. In [56], a blendshape model is built and fitted to a monocular video off-line, and then the surface detail is added by shading-based shape refinement under general lighting conditions. [46] uses a similar approach and refines the results by iteratively optimizing the large-scale facial geometry and the fine-scale facial detail. [22] proposes to create fully rigged, personalized 3D facial avatars from hand-held video. The method performs the tracking and reconstruction computation on PC though the input images are captured by mobile devices. [17] fits a 3D face using a multi-layer approach and extracts a high-fidelity parameterized 3D face rig that contains a generative wrinkle formation model capturing the person-specific idiosyncrasies. As the low-rank parametric face model limits its expressiveness and ability to capture fine details, [49] proposes to derive a person-specific face model from all available images, and each frame of the video is reconstructed by a novel 3D optical flow approach coupled with shading cues. In general, these methods work off-line and are not suitable for real-time 3D facial performance capture.

To achieve real-time capturing, [11] presents a learning-based regression approach to fit a generic identity and expression model to an RGB face video on the fly. [59] extends it to run on mobile devices at real-time frame rates by directly regressing the head poses and expression coefficients in one-step. The approach is also extended to include the learning of fine-scale facial wrinkles [10]. [54] presents a method to jointly fit a parametric model for identity, expression and skin reflectance to the input color, which achieves real-time 3D face tracking and facial reenactment while most of previous methods for facial reenactment work offline [14], [15].

CNN-based 3D Face Reconstruction from RGB Inputs. Due to the powerfulness of convolution neural networks, various deep learning methods have been developed for 3D face reconstruction, face alignment, face recognition and dense facial correspondences from RGB images. Examples are [26], [4], [62], [29], [34], [55], [25], [19], [60]. In these methods, 3D Morphable Model (3DMM) [6], [42] is used to represent 3D faces and CNN is applied to learn the 3DMM and pose parameters. For unconstrained real-time 3D facial performance capture, [44] presents a method through explicit semantic segmentation in the RGB input. By a regression-based facial tracking with segmented face images as training, uninterrupted facial performance capture can be achieved. [40], [41] present methods for 3D face reconstruction from an RGB image by training a coarse-scale CNN in a fully supervised manner using synthetic data to learn the 3DMM parameters and training a fine-scale CNN in an unsupervised manner to regress per-pixel depth displacement for detailed geometry with the first-order spherical harmonics lighting model. [31] proposes to directly learn the vertex positions from RGB images by building an initial PCA basis. [20] proposes a fully supervised deep learning framework for reconstructing a detailed 3D face from monocular RGB video in real time. Unsupervised training has been proposed by [52] using an analysis-by-synthesis energy function. [23] proposes to directly regress volumes with CNN for a single face image. While these CNN-based methods for 3D face reconstruction from RGB images can achieve impressive real-time performance, they have inherent limitations due to the RGB inputs: easily affected by illumination; unable to handle images captured in dark environment.

Facial Performance Capture from RGB-D Inputs. Compared to RGB inputs, RGB-D inputs provide additional depth information that benefits more robust facial performance capture with better quality. In fact, Weise et al. [58] developed a facial performance capture system that combines 3D and 2D non-rigid registration in optimization and achieves real-time robust 3D face tracking. This system however needs to build an accurate 3D expression blendshape by scanning and processing a predefined set of facial expressions in advance. [32] describes a system that only requires pre-building a neutral face model, generates initial blendshapes using deformation transfer [48], and then trains PCA-based correctives for the blendshapes during tracking with examples obtained from per-vertex Laplacian deformations. [7] introduces a calibration-free system that requires no user-specific preprocessing by jointly solving for a detailed 3D expression model of the user and the corresponding dynamic tracking parameters via an optimization procedure. [13] proposes an approach that incrementally deforms a 3D template mesh model to best match the input RGB-D data and facial landmarks detected from the single RGB-D image. [21] presents a realtime facial tracking system in unconstrained settings using a consumer-level RGB-D sensor. The system personalizes the tracking model on-the-fly by progressively refining the user’s identity, expressions, and texture with reliable samples and temporal filtering.

To generate high quality 3D face models from RGB-D data, [33] proposes an approach consisting of two stages: offline construction of personalized wrinkled blendshape and online 3D facial performance capturing. [53] presents a method that reconstructs high-quality facial performance of each actor by
jointly optimizing the unknown head pose, identity parameters, facial expression parameters, face albedo values, and the illumination. The method focuses on photo-realistic capture and re-rendering of facial templates to achieve expression transfer between real actors. Different from fitting the RGB-D data to the parametric model, [35] proposes to divide the input depth frame into several semantic regions and search for the best matching shape for each region in database. The reconstructed 3D mesh is constructed by combining the input depth frame with the matched shapes in the database. [37] proposes a direct approach that reconstructs and tracks the 3D face without templates or prior models.

Compared to existing real-time RGB-D based methods, which can only reconstruct a smooth 3D face in real time on PC [53] due to complex optimization, our deep learning based solution can achieve a similar quality reconstruction in real time on mobile device or a fine-scale 3D face reconstruction with geometry detail in real time on PC. This mainly owes to deep learning paradigm, which shifts the complexity to offline training while testing can be very fast. In addition, the newly embedded NPU (Neural Processing Unit) chips at mobile devices also greatly empower the computational capability of mobile devices for deep learning. Compared to existing real-time RGB based methods, our approach makes use of the captured depth information for better self-supervision during training, and can support both RGB-D and RGB inputs during testing, which leads to more reliable and accurate 3D face reconstruction.

III. FACE REPRESENTATION AND RENDERING

This section describes 3D face representation and the rendering process used in our work.

A. Face Representation

The face is basically represented as a vector \( \mathbf{p} = [\mathbf{p}_1^T, \mathbf{p}_2^T, \ldots, \mathbf{p}_n^T]^T \in \mathbb{R}^{3n} \) via a mesh of \( n \) vertices \( v_i \) \((i = 1, 2, \ldots, n)\) with fixed connectivity, where \( \mathbf{p}_i \) denotes the position of vertex \( v_i \). A parametric face model is used to specify the connectivity and geometry of the mesh, which gives a low rank representation. Fine-scale facial geometry is further added to refine the positions of vertices of the mesh.

Parametric Face Model. While many previous works use 3DMM as the parametric face model to encode 3D face geometry and albedo, we encode facial identity and expression with a bilinear face model based on FaceWareHouse [12] similar to [11], which is shown to better represent 3D face expressions. We collect the vertex coordinates of all face meshes in FaceWareHouse into a third-order data tensor and perform 2-mode SVD reduction along the identity mode and the expression mode to derive a bilinear face model that approximates the original dataset. Specifically, the facial geometry is represented as:

\[
\mathbf{p} = \mathbf{A}_r \times_2 \mathbf{\alpha}_{id} \times_3 \mathbf{\alpha}_{exp},
\]

where \( \mathbf{A}_r \) is the reduced core tensor computed from the SVD reduction and \( \mathbf{\alpha}_{id} \in \mathbb{R}^{30}, \mathbf{\alpha}_{exp} \in \mathbb{R}^{17} \) are identity and expression coefficients that control the face shape. Following [6], the facial albedo \( \mathbf{b} \) is represented with PCA:

\[
\mathbf{b} = \bar{\mathbf{b}} + \mathbf{A}_{alb} \mathbf{\alpha}_{alb},
\]

where \( \bar{\mathbf{b}} \) denotes average face albedo, \( \mathbf{A}_{alb} \) denotes the principal axes extracted from a set of textured 3D meshes, and \( \mathbf{\alpha}_{alb} \) is the albedo coefficient vector. To obtain the albedo basis, we transfer the albedo basis from Basel Face Model (BFM) [38] to the mesh representation of FaceWareHouse [12] by nonrigid registration.

Fine-Scale Facial Geometry. While the parametric representation captures global shape information of individuals, it has difficulty in handling fine-level face details such as wrinkles and folds. To alleviate this issue, we further encode fine-level facial details by per-vertex displacements along vertex normals, similar to the approach in [17].

B. Rendering Process

The rendering process of a face image depends on several factors: face geometry, albedo, pose, lighting and camera parameters. To generate a realistic synthetic image of a face, we render the facial imagery using a standard perspective pinhole camera, which can be parameterized as follows:

\[
\mathbf{q}_i = \Pi(\mathbf{R}\mathbf{p}_i + \mathbf{t})
\]

where \( \mathbf{p}_i \) and \( \mathbf{q}_i \) are the locations of vertex \( v_i \) in the world coordinate system and in the image plane, respectively, \( \mathbf{R} \) is the rotation matrix constructed from Euler angles \( pitch, yaw, roll \), \( \mathbf{t} \) is the translation vector and \( \Pi : \mathbb{R}^3 \to \mathbb{R}^2 \) is a perspective projection.

To model the lighting condition, we approximate the globe illumination using the spherical harmonics (SH) basis functions [36]. Furthermore, we assume that a face is a Lambertian surface. Under this assumption, the irradiance of a vertex \( v_i \) is determined by vertex normal \( \mathbf{n}_i \) and scalar albedo \( b_i \):

\[
\mathbf{I}(\mathbf{n}_i, b_i | \gamma) = b_i \sum_{k=1}^{3} \gamma_k \phi_k(\mathbf{n}_i) = b_i \mathbf{\phi}(\mathbf{n}_i) \cdot \gamma
\]

where \( \mathbf{\phi}(\mathbf{n}_i) = [\phi_1(\mathbf{n}_i), \ldots, \phi_3(\mathbf{n}_i)]^T \) is the SH basis computed with normal \( \mathbf{n}_i \) and \( \gamma = [\gamma_1, \ldots, \gamma_3]^T \) is the SH coefficients. In this paper, we use the first three bands \( B = 3 \) of SHs for the illumination computation.

IV. PROPOSED CNN LEARNING FRAMEWORK FOR RGB-D 3D FACE CAPTURE

The overall pipeline of the proposed CNN learning framework is shown in Fig. 2. The framework consists of two CNNs: medium-scale CNN and fine-scale CNN. The medium-scale CNN is to regress the medium-scale facial representation from an input RGB-D frame and the fine-scale CNN is to regress per-vertex displacements along vertex normals from the current medium-scale reconstruction.
Figure 2: The pipeline of our proposed CNN learning based method for RGB-D 3D Face Capture. The medium-scale CNN regresses the parametric face model and the parameters including shape, pose, lighting and albedo. The fine-scale CNN directly regresses a displacement for every vertex along the vertex normal direction in the UV map domain.

A. Training Data Preparation

To train the medium-scale CNN, we capture RGB-D videos of about 200 people with PrimeSense and about 400 people with iPhone X. The average number of frames in each video is around 500 and the total number of RGB-D training frames is about 300K. The male-to-female ratio of our training data is 1:1. About 60% are Chinese, 20% are Pakistanis/Bengalese, and 20% are Europeans. The ages of people in training data are mainly from 10 to 60. In each video, we capture the same person with different expressions (smile, frown, mouth open, etc) and poses under different lighting conditions (morning, afternoon and evening). Fig. 3 shows samples of the captured data. To train the fine-scale CNN, we only use the RGB-D videos captured with iPhoneX because of its high-quality color images, which are needed for shading-based surface detail recovery.

The input RGB-D images fed to CNN need to be cropped by the detected facial bounding boxes. Since both the crop operation and the perspective projection (relative to camera intrinsic parameters) are linear transformation, we adjust the perspective projection to match the crop operation by changing the known camera intrinsic parameters (focal length and principle point). This is analogous to the process of training images captured by different depth cameras (with different intrinsic parameters) at the same time.

For depth information input, a straightforward way is to directly feed the depth map to CNN. This, nevertheless, may make CNN difficult to learn translation due to the lack of camera intrinsic parameters. Another approach is to explicitly feed both the depth map and camera intrinsic parameters to CNN, by which the results are getting better, but they are not good enough. Our approach is to convert the depth value at every pixel to 3-channel point coordinates in the camera space with the camera intrinsic parameters, before feeding it to CNN. In this way, the network can directly infer Euler angles and the translation vector from the 3-channel point coordinates. Our experiments show that it achieves better results than the two straightforward approaches.

For RGB images, we scale each color channel with a randomly generated scalar \( s \in (0.5, 1.5) \) for data augmentation of varying lighting conditions. We also randomly disturb the cropped operation of RGB-D images for robust and efficient tracking, considering that for the sake of speedup a facial bounding box is determined based on the regressed facial parameters of the last frame rather than doing face detection at each frame. To handle occluded faces, we capture 2000 RGB-D hand images. During training process, we randomly simulate faces occluded by hands by translating the hands in front of faces. This could be easily done since we have the depth information. We then feed the simulated data into the network for training. Note that although the input are occluded faces, we use original RGB-D images as supervision when computing the loss functions. Fig. 4 shows samples of the proposed data augmentation.

B. Medium-Scale CNN

The medium-scale CNN is targeted at the medium-scale geometry. That is, given an RGB-D image as input, our medium-scale modeling network regresses
The reduced core tensor $A_r$ and the parameter set $\chi = \{\alpha_{id}, \alpha_{exp}, \alpha_{alb}, \text{pitch}, \text{yaw}, \text{roll}, t, r\}$, where $r = (\gamma_r^T, \gamma_g^T, \gamma_b^T)^T$ denotes the SH illumination coefficients of RGB channels.

The medium-scale network training architecture is shown in Fig. 5. Our training architecture has two characteristics. The first one is that we always use a pair of RGB-D frames to train the network. This enforces temporal coherence and identity consistency across individual frames. Though it seems that there are two streams in the network training architecture for each of the two RGB-D frames respectively, the two streams are actually sharing the same network.

The second characteristic is that for each RGB-D frame, the RGB channels and the 3-channel point coordinates (converted from depth) go through different networks first and their features are later concatenated for the parameter regression. Such RGB-D learning structure has been proven better than directly concatenating RGB-D data at the input [57].

Our overall loss function for training can be written as

$$E_{\text{loss}} = \frac{E_{\text{geo}} + w_\text{col} E_{\text{col}} + w_\text{lan} E_{\text{lan}} + w_\text{reg} E_{\text{reg}}}{\text{Single Loss}} + \frac{w_\text{flow} E_{\text{flow}} + w_\text{same} E_{\text{same}}}{\text{Pair Loss}}$$

where the first four terms are to measure how well the regressed parameters explain the input RGB-D observations from each frame, the last two terms $E_{\text{flow}}$ and $E_{\text{same}}$ are to measure how well the parameters of two adjacent frames match the optical flow between them and how close the identity and albedo parameters are between any two frames from the same video, respectively, and $w_{\text{col}}, w_{\text{lan}}, w_{\text{reg}}, w_{\text{flow}}$ and $w_{\text{same}}$ are tuning weights compensating for scaling of the terms. During training, we sample both pairs of adjacent frames and pairs of non-adjacent frames, where $E_{\text{flow}}$ is only applied to adjacent frames to ensure temporal smoothness and $E_{\text{same}}$ is applied to all pairs of frames so as to ensure identity consistency even across a long time span. In the following, we formulate each term of (5) in detail.

**Geometry Terms.** We use two terms to measure how well the reconstructed mesh matches the input depth frame:

$$E_{\text{geo}}(\chi) = w_{\text{pp}} E_{\text{pp}}(\chi) + w_{\text{ps}} E_{\text{ps}}(\chi)$$

where $w_{\text{pp}}$ and $w_{\text{ps}}$ are tuning weights. The first term in (6) is the point-to-point distances of back-projected 3D points between the real depth map and the rendered depth map:

$$E_{\text{pp}}(\chi) = \frac{1}{|F|} \sum_{m \in F} \|\mathbf{p}_{\text{syn}}(m) - \mathbf{p}_{\text{real}}(m')\|_2,$$

where $\| \cdot \|_2$ is the $\ell_{2,1}$ norm, $F$ is the set of all visible pixels, $\mathbf{p}_{\text{syn}}(m)$ and $\mathbf{p}_{\text{real}}(m')$ are the back-projected locations of pixel $m$ in the rendered depth map and pixel $m'$ in the real depth map, respectively, where $m'$ is the pixel in the real depth map whose back-projected location is closest to $m$'s. For computational efficiency, we search for $m'$ in a $[-5, 5] \times [-5, 5]$ neighbor region of $m$. To get the set of all visible pixels $F$, we do rasterization with the current geometry and pose. The second term in (6) evaluates the point-to-surface distance:

$$E_{\text{ps}}(\chi) = \frac{1}{|F|} \sum_{m \in F} \|\mathbf{n}_{\text{real}}(m')(\mathbf{p}_{\text{syn}}(m) - \mathbf{p}_{\text{real}}(m'))\|_2,$$

where $\mathbf{n}_{\text{real}}(m')$ is the normal of pixel $m'$ computed based on the neighboring pixels in the real depth map. It is worth pointing out that (7) and (8) are formulated in $\ell_{2,1}$ norm and $\ell_1$ norm respectively, which improve the resistance to noise and outliers that often exist in the captured depth maps. Note that the optimization of the geometry term is actually a non-rigid sparse ICP procedure. Based on the correspondence between the current 3D face mesh and the input point cloud, we update the rigid (rotation and translation) and non-rigid (identity and expression parameters) in backpropagation during training. This process is iterated until convergence.

**Color Term.** The color term evaluates how well the rendered face image based on the regressed parameters matches the input RGB image:

$$E_{\text{col}}(\chi) = \frac{1}{|F|} \sum_{m \in F} \|\mathbf{I}_{\text{syn}}(m) - \mathbf{I}_{\text{real}}(m)\|_2,$$

where $\mathbf{I}_{\text{syn}}(m)$ and $\mathbf{I}_{\text{real}}(m)$ are the synthetic color and the real color at pixel $m$, respectively. We also use $\ell_{2,1}$ norm for robustness since we do not account for specular reflection and person-specific albedo, which are considered as outliers in our model.

**Landmark Term.** The landmark term measures how close the projected vertices are to the corresponding landmarks in the input image. Moreover, for better handling expressions with eye movement, we add a term measuring the change of
the distance between upper and lower eyelids. Specifically, the landmark term is formulated as:

\[ E_{\tan}(\chi) = \frac{1}{|L|} \sum_{i \in L} \| q_i - \Pi(Rp_i + t) \|^2 + \frac{1}{|LP|} \sum_{(i,j) \in LP} \| (q_i - q_j) - \left( \Pi(Rp_i + t) - \Pi(Rp_j + t) \right) \|, \]

where \( L \) is the set of landmarks, \( q_i \) is a detected landmark position in the input image, \( p_i \) is the corresponding vertex location in the 3D mesh, \( (i,j) \) is a pair of points on the upper and lower eyelids, and \( LP \) is the set of all the eyelid pairs. We adopt the method proposed in [9] to detect facial landmarks and then manually refine the landmarks along the contour of mouth and eyes. During training, half of the contour landmark points are discarded according to the currently regressed yaw angle since the key points on the left (right) contour are invisible when people turn left (right).

**Regularization Term.** The regularization term aims to ensure that the parameters of the regressed parametric face model are plausible:

\[ E_{\text{reg}}(\chi) = \frac{50}{|s|} \sum_{i=1}^{50} \left( \frac{\alpha_{id,i}}{\sigma_{id,i}} \right)^2 + \frac{50}{|s|} \sum_{i=1}^{50} \left( \frac{\alpha_{alb,i}}{\sigma_{alb,i}} \right)^2 + \frac{47}{|s|} \sum_{i=1}^{47} \left( \frac{\alpha_{\exp,i}}{\sigma_{\exp,i}} \right)^2, \]

where \( \sigma \) are the standard deviations of the corresponding principal directions. In this work, we use 50 principal components for albedo, 50 for identity and 47 for expression.

**Temporal Coherence Term.** Note that the above four terms are frame-to-frame independent. To account for temporal coherence between frames for stable tracking, additional terms are needed. To this end, we introduce a temporal coherence term that measures how well the moving flow of the projected mesh matches the optical flow between two adjacent RGB frames:

\[ E_{\text{flow}}(\chi_n, \chi_{n-1}) = \frac{1}{|F|} \sum_{m \in F} \| PR_n(p) - PR_{n-1}(p) - f(m) \|^2, \]

where \( \chi_n \) and \( \chi_{n-1} \) are the regressed parameters for the current frame and the previous frame respectively, \( p \) is the corresponding position on the 3D mesh for pixel \( m \) by barycentric interpolation of the three vertices of the underlying triangle, \( PR_n(p) \) and \( PR_{n-1}(p) \) are the projections of \( p \) on the image plane for the current frame and the previous frame respectively, and \( f(m) \) is the optical flow at pixel \( m \). We compute the dense optical flow using the method of [61]. Note that we compute optical flow before data augmentation, and thus we can obtain accurate optical flow for the augmented dark facial images.

**Same Identity Term.** When tracking the same person, identical identity and albedo parameters for different frames are expected. However, since identity and expression are not totally independent in the parametric face model, the identity parameters of the same person at different frames might vary during the tracking process. To alleviate this problem, we include an identity term to ensure the consistency of the identity and albedo parameters of the same person:

\[ E_{\text{same}}(\chi_{n_1}, \chi_{n_2}) = \left( \| \alpha_{id,n_1} - \alpha_{id,n_2} \|_2^2 + \| \alpha_{alb,n_1} - \alpha_{alb,n_2} \|_2^2 \right), \]

where \( n_1 \) and \( n_2 \) are any two frames of the same person, including non-adjacent frames.

**Learning Bilinear Face Model.** Note that the bilinear face model \( A_r \) defined in (1) is constructed by performing 2-mode SVD reduction from the FaceWareHouse dataset [12]. Although the SVD reduction is a good way to extract the low-dimensional facial model, it is not necessarily optimal. Moreover, it may not work well for faces that do not belong to the types included in the FaceWareHouse dataset. For this reason, we propose not to fix the reduced core tensor, but to further refine it with our collected large-scale training dataset. Our strategy is to use a dictionary learning process [1], where estimating the parameters of the parametric face model is a coding step and refining the core tensor is a dictionary update step. Specifically, we denote by \( A_r' \), the refined core tensor that is assumed to better represent the middle-level geometry, and add the following term to (5):

\[ E_{\text{reg}}(A_r') = w_{\text{reg}} \| A_r' - A_r \|^2 + w_{\text{amo}} \| \Delta(A_r' - A_r) \|^2, \]

where \( \Delta(A_r' - A_r) \) is the Laplacian computed on a mesh of \( 3 \times 50 \times 47 \) dimensions, and \( w_{\text{reg}} \) and \( w_{\text{amo}} \) are tuning weights. In particular, we create a 'geometry' layer before the loss layer, where we set \( A_r' \) as the 'weights' (learnable parameters), identity and expression parameters as the input and the output is the reconstructed 3D geometry. When learning \( A_r' \) from the amended (5), we keep parameters \( \chi \) fixed (set learning rate to zero for convolutional layers) and update \( A_r' \) using Stochastic Gradient Descent (SGD).

**Remarks:** We would like to point out that the first four terms in (5) are in principle similar to those used in the optimization-based method [53]. However, there are a few fundamental differences. First, we use either \( \ell_{2,1} \) norm or \( \ell_1 \) norm for the first two terms in (5), while [53] uses \( \ell_2 \) norm in order to achieve the real-time requirement. Using \( \ell_{2,1} \) norm and \( \ell_1 \) norm makes our method more robust to the input depth noise and the imperfection of the parametric facial model. Second, although [53] jointly optimizes facial identity and reflectance on a short control sequence, the temporal coherence between adjacent frames (e.g. optical flow) is not considered. We add the temporal coherence constraint via the pair loss which helps to improve the tracking stability. The last but most important one is that we refine the core tensor during CNN training in addition to regressing the model parameters, which greatly improves the reconstruction accuracy. Moreover, the key insight here is that our solution is a learning based approach, which can afford to use more sophisticated loss terms during offline training without affecting the real-time performance during online inference. In contrast, the existing optimization-based real-time RGB-D solutions can only use simple terms and less constraints in order to achieve the real-time requirement, not to mention that they do not reconstruct fine-scale 3D facial details.

**C. Fine-Scale CNN**

The purpose of the fine-scale CNN is to refine the surface details of the medium-scale face mesh obtained from the medium-scale CNN. [31] proposes a CNN framework to
directly learn vertex positions from RGB images, but the method needs ground-truth vertex positions for fully supervised learning and a per-user calibration for building an initial PCA basis, which needs to be retrained for each new identity. In addition, using a fully connected layer to regress all vertex positions in [31] is neither efficient nor reasonable, considering that there is a large number of vertices (say, 47954 vertices) and each vertex is only dependent of its neighbor vertices. [41] and [20] regress a per-pixel depth displacement map in an unsupervised and supervised manner, respectively, to refine facial surface details. Although the representation of the per-pixel depth displacement map is quite convenient for CNN learning, it does not align with the 3D face mesh representation and it only refines the geometry of a particular view.

In this work, we make use of shading information, which is related to the surface normal whose variance intrinsically reflects the local high-frequency shape changes, to learn a displacement for every vertex along the vertex normal direction for fine-scale face modeling. Considering that the conventional CNN learning prefers inputs and outputs in a regular grid, we parameterize the 3D face mesh to a UV map, which is done by minimizing the symmetric Dirichlet energy [47], and regress the displacements in the UV map domain. Specifically, the input to the fine-scale CNN is a four-dimensional map in the UV domain, where each UV pixel contains the intensity value and the XYZ coordinates of the corresponding 3D point on the reconstructed medium-level 3D face mesh. The output to the fine-scale CNN is another UV map, where each UV pixel contains one scalar value representing the predicted normal-direction displacement for the corresponding 3D point on the face mesh. In this way, we can use a pixel-to-pixel network structure, which is more efficient and better preserves local information. Fig. 6 gives examples of the input intensity image and the XYZ image parameterized on the UV domain, where the resolution of UV map is 450 × 450.

We denote by \( \mathbf{d} \) the output UV map containing the displacements of all pixels and we define our loss function for fine-scale CNN based on shape from shading. The function is a sum of a shading term \( E_{sh}(\mathbf{d}) \), a regularization term \( E_{sm}(\mathbf{d}) \) and a temporal coherence term \( E_{cl}(\mathbf{d}_n, \mathbf{d}_{n-1}) \), each of which is explained below.

**Shading Term.** Considering that in a triangular mesh a face has well-defined normal compared to vertices, we define our shading term based on faces rather than vertices:

\[
E_{sh}(\mathbf{d}) = w_{face} \sum_{i \in T} \| \mathbf{I}(\mathbf{n}_i, b_i, \gamma) - \mathbf{c}_i \|^2 + w_{edge} \sum_{i \in T} \left( \sum_{j \in E_i} \| \mathbf{I}(\mathbf{n}_i, b_i, \gamma) - \mathbf{I}(\mathbf{n}_j, b_j, \gamma) \| - (\mathbf{c}_i - \mathbf{c}_j) \|^2 \right),
\]

where \( w_{face}, w_{edge} \) are tuning weights, \( T \) is the set of all triangles in the 3D face mesh, \( E_i \) is the set of all edges, \( \mathbf{c}_i \) is the intensity value at the center of triangle \( i \) sampled by projecting the 3D face mesh onto the image plane and bilinearly interpolating neighbor pixels, and \( \mathbf{I}(\mathbf{n}_i, b_i, \gamma) \) is the predicted intensity with the current normal \( \mathbf{n}_i \) given the obtained albedo \( b_i \) and the SH lighting coefficients \( \gamma \). Note that \( \mathbf{n}_i \) is indirectly affected by the UV displacement map \( \mathbf{d} \). That is, given the fine-scale CNN output \( \mathbf{d} \), we update the vertex positions of the 3D face mesh accordingly, based on which we then update triangle normals \( \mathbf{n}_i \) and compute (15).

**Regularization Term.** We regularize the face mesh to be smooth and close to the reconstructed mesh in medium-scale level by minimizing the \( \ell_2 \) norm of Laplacian vectors and the \( \ell_2 \) norm of \( \mathbf{d} \) respectively:

\[
E_{sm}(\mathbf{d}) = w_{sm} \sum_{v_i \in V} \| \mathbf{p}_i \|_2 + w_{sm} \| \mathbf{d} \|_2^2,
\]

where \( w_{sm} \) and \( w_{sm} \) are tuning weights, \( V \) is the set of all vertices, and \( \mathbf{p}_i \) is the position of vertex \( v_i \).

**Temporal Coherence Term.** Similar to the medium-scale face modeling, we also add a temporal coherence term that measures the difference of triangle normals between two adjacent frames:

\[
E_{cl}(\mathbf{d}_n, \mathbf{d}_{n-1}) = w_{cl} \sum_{i \in T} \| \mathbf{n}_{n,i} - \mathbf{n}_{n-1,i} \|_2^2,
\]

where \( w_{cl} \) is a tuning weight, and \( \mathbf{n}_{n,i} \) and \( \mathbf{n}_{n-1,i} \) are the unit normals of triangle \( i \) in the current frame and the previous frame, respectively.

### D. Extensions to RGB Based Facial Performance Capture

It is worth pointing out that our CNN framework is very flexible. Though it is designed for facial performance capture with an RGB-D camera, we can easily extend it to facial performance capture with only RGB input. The novel idea here is that to use both RGB and depth data during training while using only RGB data during testing. Specifically, during training, we use RGB data as input to match the testing scenario while using the additional depth information as supervision at the output, i.e. keeping similar RGB-D terms in the loss function. This is very different from the existing deep learning based 3D face reconstruction methods from RGB images, where only RGB images are used in both training and testing. Our
experiments show that the addition supervision from depth during training delivers better reconstruction results.

Particularly, we train a model with only RGB images as input but still keep the depth related terms in the loss function to guide the CNN model. Our initial attempt is to adopt the same loss function as that in the scenario with RGB-D input. However, the loss of the geometry terms cannot converge since it is not easy to accurately estimate the real-world translation through CNN with only RGB image input. Therefore, rather than using the point-to-point distance and the point-to-surface distance in (6), we use the normal-to-normal distance between the real depth map and the rendered depth map, which is defined as:

$$E_n(\chi) = \frac{1}{|\mathcal{F}|} \sum_{m \in \mathcal{F}} \|n_{\text{syn}}(m) - n_{\text{real}}(m')\|_2,$$  \hspace{1cm}  (18)

where $\| \cdot \|_2$ is the $\ell_{2,1}$ norm, $\mathcal{F}$ is the set of all visible pixels, $m'$ is the pixel in the real depth map whose back-projected location is closest to $m$, and $n_{\text{syn}}(m)$ and $n_{\text{real}}(m')$ are the unit normals of the rendered depth map at pixel $m$ and the real depth map at pixel $m'$, respectively.

V. RESULTS

In this section, we first evaluate the performance of our learning based 3D face capture method in different aspects, and then compare it with state-of-the-art methods.

Implementation. We train our CNNs based on Caffe [24]. For training the medium-scale neural network, we first use the landmark term and the regularizarion term to do an initial alignment, and then add other terms for a dense alignment. When feeding the input image into the network, we crop the face region to $224 \times 224$ with bilinear interpolation. In the testing process, the face region of the first frame is generated by face detection, and the bounding box of any other frame is automatically obtained according to the predicted landmarks by projecting the reconstructed 3D mesh of the previous frame. For the medium-scale CNN, we use GoogLeNet [50] structure and concatenate the RGB features and XfZ features after the third inception module. For the fine-scale CNN, we use U-Net [43] structure. We train our networks using Adam solver with the mini-batch size of 50 for both networks. We train the medium-scale CNN by iteratively updating the convolutional layers with 20K iterations and the parametric model $A_r$ with 20K iterations, and the total number of iterations is 200K. The fine-scale CNN is trained with 30K iterations. The weights for loss functions including $w_{\text{fps}}, w_{\text{ps}}, w_{\text{col}}, w_{\text{lan}}, w_{\text{reg}}, w_{\text{flow}}, w_{\text{same}}, w_{\text{Avg}}, w_{\text{Asym}}, w_{\text{face}}$, $w_{\text{edge}}, w_{\text{sm}}, w_{\text{mi}}$ and $w_{\text{cl}}$ are empirically set to 2, 2, 1, 0.5, 0.5, 0.5, 0.0001, 1000, 2000, 0.1, 0.1, 0.1, 0.02 and 0.01, respectively.

Our CNN based 3D face reconstruction and tracking are implemented in C++ and tested on various RGB-D sequences. We conduct experiments on a desktop PC with a quad-core Intel CPU i7, 4GB RAM and NVIDIA GTX 1070 GPU. Besides, we also export our trained medium-scale CNN to HUAWEI Mate 10 mobile device, which is with Kirin 970 NPU (FP16 1.92 TFLOP) and Octa-core CPU (4x2.4 GHz Cortex-A73 & 4x1.8 GHz Cortex-A53). As for the running time for each frame, on the PC, it takes 9ms for medium-scale CNN and 14ms for fine-scale CNN. On HUAWEI Mate 10 smartphone, on average our medium-scale CNN takes around 33 ms for one frame with half-precision float. Note that our fine-scale CNN cannot achieve real-time computation on the smartphone, but on the common PC we demonstrate that the combination of both medium-scale CNN and fine-scale CNN can be run in real time. Since this paper focuses on real-time 3D face reconstruction and tracking on mobile devices, in the following without specification the results of our method refer to the results of our medium-scale CNN, which can be run on mobile devices in real time.

A. Evaluation of Our Method

Robustness. Fig. 7 shows the reconstructed 3D faces of our medium-scale CNN model on sampled RGB-D frames with large poses, different expressions, motion blur, different lighting conditions and occlusions. The accompanying video shows the tracking results of the complete sequences. With the additional depth information, our method is more robust than the RGB based methods. As shown in Fig. 8, the state-of-the-art RGB-based methods cannot perform well or even fail to reconstruct the 3D face with the RGB images captured in dark environment, while our method with RGB-D input can still reconstruct the 3D face shape well thanks to the additional depth information.

The reconstruction result comparison between with and without occlusion data augmentation is shown in Fig. 9, and it shows that the mouth part is distorted if the training data set does not include occlusions, while our method could achieve better results. From these experiments, we can observe that our method is robust to these challenging scenarios, which is mainly due to a diversity of identities, poses, expressions and lighting conditions contained in our large-scale training data set. To further improve the robustness, we also do data augmentation including random cropping and scaling, flipping and occlusion simulation during training. With these diverse data and our carefully designed training strategies, we successfully train a powerful and robust model offline, which can then be run in real time during the inference stage. Note that Thies et al. [53] cannot handle very fast head motion and dark environments, as pointed out in their paper.

Generalization Ability. Although our medium-scale CNN model is trained with the RGB-D data captured by PrimeSense camera and iPhoneX, it can be applied to other RGB-D cameras. We test our trained medium-scale CNN model with different depth cameras including Kinect (V1), PrimeSene, Intel RealSense F200, and iPhone-X. Fig. 10 shows the 3D reconstruction results with different sensors on the same person. It can be observed that the reconstruction results across different sensors are consistent.

Temporal Coherence. We demonstrate the advantage of the temporal coherence term in (12) in the medium-scale CNN training by comparing the tracking results between the models trained with and without the temporal coherence term. We apply both methods on an RGB-D sequence containing 100 frames of a face (kept as static as possible) and evaluate the
Figure 7: Reconstruction results of our medium-scale CNN model on sampled RGB-D frames with large poses, different expressions, motion blur, different lighting conditions and occlusions. For every three rows from top to bottom: input face images, reconstructed 3D meshes overlayed with input point clouds, reconstructed 3D meshes overlayed with input images.

The average vertex displacements of each frame w.r.t the mean 3D face averaged over all the reconstructions. The average vertex displacements of each frame are shown in Fig. 11. It can be observed that the reconstruction results with the temporal coherence term are more stable than those without the temporal coherence term.

Reconstruction Quality of Medium-scale CNN. Fig. 12 gives a few examples of the reconstructed 3D meshes and the corresponding geometry fitting errors. Geometry fitting error is defined as the point-to-point distance between the reconstructed 3D mesh and the input point cloud. Here, we show the fitting results of not only our medium-scale CNN model but also a trained model that is almost the same as the medium-scale CNN model except that it does not optimize the tensor $A_r'$. We test our method on an RGB-D sequence including 150 frames, and the mean and standard deviation values of the geometry fitting errors are 2.33 mm / 0.45 mm and 1.91 mm / 0.23 mm for the models without and with optimizing $A_r'$, respectively. Fig. 12 shows three frames of the RGB-D sequence, and we can observe that the geometry fitting errors are reduced significantly with the help of $A_r'$. We also evaluate our method on a large-scale test set including 152 people and 73580 RGB-D images, and the mean and standard deviation values of the geometry fitting errors between our reconstructed results and the input point clouds are 1.83mm / 0.74mm.

Our method also works well for subtle expressions like mouth and eye movements, as shown in Fig. 13. This is due to
two reasons. First, we learn to refine the bilinear face tensor to better fit the scanned RGB-D data, which significantly improves the expression ability. Second, we add the distance constraint in Eq. 10 to improve the shape on the ‘eye’ parts.

**Reconstruction Quality of Fine-scale CNN.** Fig. 14 compares the 3D reconstruction results of our medium-scale model and our fine-scale model. It can be seen that compared to the medium-scale CNN, the fine-scale CNN recovers more fine geometric details such as wrinkles. This is because our fine-scale CNN is able to directly add the geometry details to the mesh vertices via regressing the displacements along each vertex normal according to shape-from-shading with the second-order SH lighting model.

**B. Comparisons with State-of-the-Art Approaches**

In this subsection, we compare our trained medium-scale CNN model with the following state-of-the-art approaches which can also reconstruct 3D faces from an RGB-D camera in real time.

[53]. - This method achieves the state-of-the-art performance on real-time 3D face reconstruction and tracking from a single RGB-D camera. It basically recovers 3D face geometry, albedo
and lighting from an RGB-D sequence simultaneously by solving an optimization problem. It achieves real-time tracking on a desktop PC with the help of GPU computing. Fig. 15 shows some reconstruction results, and the full comparisons on all the video frames are given in the accompanying video. Our results are comparable to them, while our method can run on mobile device in real time. Note that we only have the video sequence used in [53], but not the reconstructed 3D meshes. Thus we cannot compare the geometry fitting errors with [53]. For the video sequence of 600 frames (see the supplementary video), our method achieves the geometry fitting errors with mean / standard deviation of 2.29 mm/0.20 mm, which are comparable to the final results of 2.26 mm/0.27 mm reported in [53].

[11]. - This method learns the 3D face shape parameters from a single RGB video stream. We compare our method with it on the test examples used in [53]. While our method takes as input the RGB-D data and the camera parameters, the method of [11] only uses the RGB data to regress the shape parameters and the camera parameter. As shown in Fig. 15, our method outputs better shapes especially for the frames with expressions.

ARKit. - Apple Inc recently released ARKit [2], which has the feature of capturing a user’s 3D face with iPhone X smartphone. We compare our reconstruction results with those obtained by the Face Mesh tool in MeasureKit 2, where all face data comes directly from ARKit. Both methods take the RGB-D data scanned by iPhone X as input. We quantitatively evaluate the geometry accuracy of both methods, which is defined as the point-to-point distance between a reconstruction mesh and the corresponding ground-truth geometry captured by Range 7 3D Scanner [30]. Note that we only compare the facial part since the laser scanner fails to capture hair. To align the reconstructed 3D face model with the ground-truth 3D point cloud, we first manually label several landmarks to do rigid registration and then apply dense ICP for dense registration. From Fig. 16, we can see that the geometry produced by our method is visually comparable or better than ARKit. The fitting errors in mean and standard deviation of our method and ARKit against the ground-truth geometry are also given for the examples in Fig. 16, where our method obtains smaller fitting errors than ARKit.

We also compare our method with ARKit for the scenario with only RGB input during testing, where we call ARKit on iPhone-X with only RGB information by disabling the depth camera, and apply it on the same examples in Fig. 16. Our method with RGB inputs gives fitting errors in mean/standard deviation: 1.72/1.46 mm, 1.83/1.89 mm and 2.02/1.58 mm respectively for the three examples in Fig. 16, while ARKit with RGB inputs gives errors: 1.81/1.69 mm, 2.32/2.12 mm and 2.27/2.17 mm.

Other RGB-based methods. - We also compare our method with the state-of-the-art RGB-based methods [23], [45] by using the source codes provided by the authors. These RGB-based methods only use RGB information, while our method takes the additional depth information, which is used only in training but not in testing, for better self-supervised learning. We compute the fitting error by aligning the reconstructed meshes with the scanned point cloud. It can be observed from Fig. 17 that our method achieves the best reconstruction quality due to the additional depth supervision. These examples demonstrate that even rough depth data captured by consumer-grade depth cameras can significantly improve the face capture performance.

---

2https://measurekit.com/
VI. DISCUSSION & CONCLUSION

We have presented an RGB-D deep learning method for 3D facial performance capture, which is able to run on mobile devices and achieves high-quality, robust, and real-time performance even under the scenarios with large poses, different facial expressions, motion blur and diverse lighting conditions. The underlying technique is a self-supervised CNN framework that trains a medium-scale CNN and a fine-scale CNN directly from RGB-D raw face data. The framework is flexible to support facial performance capture with RGB-D or RGB input during testing. The key technical novelties lie in the training of the parametric face model, which learns both the core tensor and the model parameters, the use of UV map and vertex displacement for CNN-based surface detail refinement, and the elaborately-designed loss function. In particular, considerable efforts have been put into devising the terms for the loss functions and integrating various components to achieve efficient and robust training of the networks. Experiments show that our method is suitable for consumer-level RGB-D sensors and capturing 3D faces in the wild, which is of great practical value.

While ARKit [2] can real-time construct 3D face models from RGB-D data captured by iPhone X smartphone, its underlying technique has not been made public. Most previous work has difficulty to achieve accurate and robust real-time 3D facial capture from RGB-D data on mobile phones. Our work suggests that using RGB-D data and deep learning techniques provides a promising solution, especially noticing that some newly-released smartphones are equipped with advanced chip (for example, Huawei’s Kirin 970 NPU) supporting deep learning.

One limitation of our current work is that while our medium-scale CNN can achieve real time facial performance capture on mobile devices, our fine-scale network still cannot due to its complexity. Optimizing the implementation and smartly modeling the surface fine detail will help speed up. Another limitation of our work is that we assume Lambertian surface reflectance and smoothly varying illumination, which may introduce artifacts in general environments (e.g., with strong subsurface scattering, high frequency or self-shadowing). Though our sparse approximation to the color information could alleviate these artifacts to a certain extent, it is worth investigating more powerful formulation to handle general reflectance and illumination.

Figure 16: Comparisons with ARKit. 1st column: ground-truth geometry captured by Range 7 3D Scanner and input images. 2nd column: results of ARKit with only RGB input and geometry accuracy (w.r.t. ground-truth geometry). 3rd column: results of ARKit with RGB-D input. 4th column: results of our method with RGB input. 5th column: results of our method with RGB-D input. The error map images show the fitting error between the reconstructed models and the groundtruth models. The mean / standard deviations of errors (mm) are listed at the bottom.

Figure 17: Comparisons with the RGB-based methods [23], [45]. From top to bottom: input facial images and scanned point clouds, results of [45], results of [23], results of ours (RGB) and results of ours (RGB-D input). The mean/std error to scanned point clouds is shown on the bottom of each error map. Note that [23], [45] use only RGB information while our method (RGB) also uses depth information in training although not in testing.
ACKNOWLEDGMENTS

We thank Thomas Vetter et al. and Kun Zhou et al. for allowing us to use their 3D face datasets. This work was supported by the National Key R&D Program of China (No. 2016YFC0800501), the National Natural Science Foundation of China (No. 61672481), and the Youth Innovation Promotion Association of CAS.

REFERENCES

[1] M. Aharon, M. Elad, and A. Bruckstein. Svd: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on Signal Processing, 54(11):4511–4322, 2006.

[2] Apple Inc. Arik, 2017.

[3] T. Beeler, F. Hahn, D. Bradley, B. Bickel, P. A. Beardsley, C. Gotsman, R. W. Sumner, and M. H. Gross. High-quality passive facial performance capture using anchor frames. ACM Trans. Graph., 30(4):75:1–75:10, 2011.

[4] C. Bhagavatula, C. Zhu, K. Lau, and M. Savvides. Faster than real-time facial alignment: A 3d spatial tracker. In Unconstrained poses. In IEEE International Conference on Computer Vision, pages 4000–4009, 2017.

[5] B. Bickel, M. Botsch, R. Angst, W. Matusik, M. Otaduy, H. Pfister, and M. Gross. Multi-scale capture of facial geometry and motion. ACM Trans. Graph., 26(3), 2007.

[6] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ’99, pages 187–194, 1999.

[7] V. Blanz, T. Vetter, F. A. Chini, S. W. Savvides, A. Bruckstein, and M. Gross. Video face replacement. ACM Trans. Graph., 29(4):112:1–112:10, 2010.

[8] A. Bulat and G. Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In IEEE International Conference on Computer Vision, 2017.

[9] C. Cao, D. Bradley, K. Zhou, and T. Beeler. Real-time high-fidelity facial performance capture. ACM Transactions on Graphics (TOG), 34(4):46:1–46:15, 2015.

[10] C. Cao, Q. Hou, and K. Zhou. Displaced dynamic expression regression for real-time facial tracking and animation. ACM Transactions on Graphics (TOG), 33(4):43, 2014.

[11] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. Facewarehouse: A 3d facial expression database for visual computing. IEEE Transactions on Visualization and Computer Graphics, 20(9):431–425, 2014.

[12] Y.-L. Chen, H.-T. Wu, F. Shi, X. Tong, and J. Chai. Accurate and robust 3d facial capture using a single rgbd camera. In IEEE International Conference on Computer Vision, pages 3615–3622, 2013.

[13] I. Kemelmacher-Shlizerman and L. G. Shapiro. 3d face hallucination from a single depth frame. In 2nd International Conference on 3D Vision, 3DV 2014, Tokyo, Japan, December 8-11, 2014, Volume 1, pages 31–38, 2014.

[14] C. Muller. Spherical harmonics. In Lecture Notes in Mathematics, volume 17, 1966.

[15] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In IEEE Conference on Computer Vision and Pattern Recognition, pages 343–352, 2015.

[16] S. Bouaziz, Y. Wang, and M. Pauly. Online modeling for real-time dense face reconstruction with inverse-rendered photo-realistic face animation. ACM Transactions on Graphics (TOG), 34(4):4, 2015.

[17] C. M¨uller. Spherical harmonics. In Lecture Notes in Mathematics, volume 17, 1966.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on Multimedia, pages 675–678. ACM, 2014.

[19] X. Yin and X. Tan. Face alignment in-the-wild: A survey. Computer Vision and Image Understanding, 162:1–22, 2017.

[20] M. Gross. Multi-scale capture of facial geometry and motion. ACM Trans. Graph., 30(2):58:117–122, 2015.

[21] S. Li, W. Xu, Z. Cheng, K. Xu, and R. Klein. Lightweight wrinkle synthesis for 3d facial modeling and animation. Computer-Aided Design, 58:117–122, 2015.

[22] M. Li, W. Zuo, and D. Zhang. Convolutional network for attribute-driven and identity-preserving human face generation. CoRR, abs/1608.06434, 2016.

[23] G. Tzimiropoulos, I. Kemenlimacher, and G. Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In IEEE International Conference on Computer Vision, 2017.

[24] C. Muller. Spherical harmonics. In Lecture Notes in Mathematics, volume 17, 1966.

[25] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In IEEE Conference on Computer Vision and Pattern Recognition, pages 343–352, 2015.

[26] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In IEEE Conference on Computer Vision and Pattern Recognition, pages 343–352, 2015.

[27] I. Kemelmacher-Shlizerman and R. Basri. 3d face reconstruction from a single image using a single reference face shape. IEEE Trans. Pattern Anal. Mach. Intell., 33(2):394–405, 2011.

[28] I. Kemelmacher-Shlizerman and S. M. Seitz. Face reconstruction in the wild. In IEEE International Conference on Computer Vision, pages 1746–1753, 2011.

[29] H. Kim, M. Zollhofer, A. Tewari, J. Thies, C. Richardt, and C. Theobalt. InverseFaceNet: Deep monocular inverse face rendering. In IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[30] S. Bouaziz, Y. Wang, and M. Pauly. Online modeling for real-time facial animation. ACM Transactions on Graphics (TOG), 32(4):40:1–40:10, 2013.

[31] T. Beeler, F. Hahn, D. Bradley, B. Bickel, P. A. Beardsley, C. Gotsman, R. W. Sumner, and M. H. Gross. High-quality passive facial performance capture using anchor frames. ACM Trans. Graph., 30(4):75:1–75:10, 2011.
[48] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics (TOG)*, 23(3):399–405, 2004.

[49] S. Suwajanakorn, I. Kemelmacher-Shlizerman, and S. M. Seitz. Total moving face reconstruction. In *ECCV*, pages 796–812, 2014.

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[51] A. Tewari, M. Zollhöfer, P. Garrido, F. Bernard, H. Kim, P. Pérez, and C. Theobalt. Self-supervised multi-level face model learning for monocular reconstruction at over 250 hz. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[52] A. Tewari, M. Zollhöfer, H. Kim, P. Garrido, F. Bernard, P. Perez, and T. Christian. MoFA: Model-based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction. In *IEEE International Conference on Computer Vision*, 2017.

[53] J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt. Real-time expression transfer for facial reenactment. *ACM Transactions on Graphics (TOG)*, 34(6):183:1–183:14, 2015.

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

[55] A. T. Tran, T. Hassner, I. Masi, and G. G. Medioni. Regressing robust and discriminative 3d morphable models with a very deep neural network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1493–1502, 2017.

[56] L. Valgaerts, C. Wu, A. Bruhn, H. Seidel, and C. Theobalt. Lightweight binocular facial performance capture under uncontrolled lighting. *ACM Transactions on graphics (TOG)*, 31(6):187:1–187:11, 2012.

[57] A. Wang, J. Cai, J. Lu, and T.-J. Cham. Mmss: Multi-modal sharable and specific feature learning for rgb-d object recognition. In *IEEE International Conference on Computer Vision*, pages 1125–1133, 2015.

[58] T. Weise, S. Bouaziz, H. Li, and M. Pauly. Realtime performance-based facial animation. In *ACM transactions on graphics (TOG)*, volume 30, page 77. ACM, 2011.

[59] Y. Weng, C. Cao, Q. Hou, and K. Zhou. Real-time facial animation on mobile devices. *Graphical Models*, 76(3):172–179, 2014.

[60] R. Yu, S. Saito, H. Li, D. Ceylan, and H. Li. Learning dense facial correspondences in unconstrained images. In *IEEE International Conference on Computer Vision*, pages 4733–4742, 2017.

[61] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-L1 optical flow. In *Pattern Recognition, 29th DAGM Symposium*, pages 214–223, 2007.

[62] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li. Face alignment across large poses: A 3d solution. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 146–155, 2016.

[63] M. Zollhöfer, J. Thies, P. Garrido, D. Bradley, T. Beeler, P. Pérez, M. Stamminger, M. Nießner, and C. Theobalt. State of the art on monocular 3d face reconstruction, tracking, and applications. *Comput. Graph. Forum*, 37(2):523–550, 2018.