CNN-Based Real-Time Dense Face Reconstruction with Inverse-Rendered Photo-Realistic Face Images

Yudong Guo, Juyong Zhang, Jianfei Cai, *Senior Member, IEEE*, Boyi Jiang, and Jianmin Zheng

Abstract—With the powerfulness of convolution neural networks (CNN), CNN based face reconstruction has recently shown promising performance in reconstructing detailed face shape from 2D face images. The success of CNN-based methods relies on a large number of labeled data. The state-of-the-art synthesizes such data using a coarse morphable face model, which however has difficulty to generate detailed photo-realistic images of faces (with wrinkles). This paper presents a novel face data generation method. Specifically, we render a large number of photo-realistic face images with different attributes based on inverse rendering. Furthermore, we construct a fine-detailed face image dataset by transferring different scales of details from one image to another. We also construct a large number of video-type adjacent frame pairs by simulating the distribution of real video data.[1] With these nicely constructed datasets, we propose a coarse-to-fine learning framework consisting of three convolutional networks. The networks are trained for real-time detailed 3D face reconstruction from monocular video as well as from a single image. Extensive experimental results demonstrate that our framework can produce high-quality reconstruction but with much less computation time compared to the state-of-the-art. Moreover, our method is robust to pose, expression and lighting due to the diversity of data.

Index Terms—3D face reconstruction, face tracking, face performance capturing, 3D face dataset, image synthesis, deep learning

1 Introduction

This paper considers the problem of dense 3D face reconstruction from monocular video as well as from a single face image. Single-image based 3D face reconstruction can be considered as a special case of video based reconstruction. It also plays an essential role. Actually image-based 3D face reconstruction itself is a fundamental problem in computer vision and graphics, and has many applications such as face recognition [5], [54] and face animation [23], [53]. Video-based dense face reconstruction and tracking or facial performance capturing has a long history [57] also with many applications such as facial expression transfer [52], [53] and face replacement [12], [16], [30]. Traditional facial performance capture methods usually require complex hardware and significant user intervention [21], [57] to achieve a sufficient reality and therefore are not suitable for consumer-level applications. Commodity RGB-D camera based methods [6], [33], [52], [56] have demonstrated real-time reconstruction and animation results. However, RGB-D devices, such as Microsoft's Kinect, are still not that common and not of high resolution, compared to RGB devices.

Recently, several approaches have been proposed for RGB video based facial performance captureing [7], [8], [18], [22], [45], [53]. Compared to image-based 3D face reconstruction that is considered as an ill-pose and challenging task due to the ambiguities caused by insufficient information conveyed in 2D images, video-based 3D reconstruction and tracking is even more challenging especially when the reconstruction is required to be real-time, fine-detailed and robust to pose, facial expression, lighting, etc. These proposed approaches only partially comply with the requirements. For example, [8] and [7] learn facial geometry while not recovering facial appearance property, such as albedo. [18] can reconstruct personalized face rig of high-quality, but their optimization-based method is time-consuming and needs about 3 minutes per frame. [53] achieves real-time face reconstruction and facial reenactment through data-parallel optimization strategy, but their method cannot recover fine-scale details such as wrinkles and also requires facial landmark inputs.

In this paper, we present a solution to tackle all these problems by utilizing the powerfulness of convolutional neural networks (CNN). CNN based approaches have been proposed for face reconstruction from a single image [24], [41], [42], [51], [54], but CNN is rarely explored for video-based dense face reconstruction and tracking, especially for real-time reconstruction. Inspired by the state-of-the-art single-image based face reconstruction method [42], which employs two cascaded CNNs (coarse-layer CNN and fine-layer CNN) to reconstruct a detailed 3D facial surface from a single image, we develop a dense face reconstruction and tracking

1. All these coarse-scale and fine-scale photo-realistic face image datasets can be downloaded from https://github.com/Juyong/3DFace.

- *Y. Guo, J. Zhang, and B. Jiang are with School of Mathematical Sciences, University of Science and Technology of China, Hefei 230000, China. E-mail: {gyd2011, jby1993}@mail.ustc.edu.cn, juyong@ustc.edu.cn.*
- *J. Cai and J. Zheng are with School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798. E-mail: {asjfcai, asjmzheng}@ntu.edu.sg.*

Manuscript received 11 Sept. 2017; revised 10 May 2018; accepted 14 May 2018. Date of publication 16 May 2018; date of current version 14 May 2019. (Corresponding author: Juyong Zhang.)
Recommended for acceptance by L. Agapito.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2018.2837742

framework. The framework includes a new network architecture called 3DFaceNet for online real-time dense face reconstruction from monocular video (supporting a single-image input as well), and optimization-based inverse rendering for offline generating large-scale training datasets.

In particular, our proposed 3DFaceNet consists of three convolutional networks: a coarse-scale single-image network (named Single-image CoarseNet for the first frame or the single image case), a coarse-scale tracking network (Tracking CoarseNet) and a fine-scale network (FineNet). For single-image based reconstruction, compared with [42], the key uniqueness of our framework lies in the photo-realistic datasets we generate for training CoarseNet and FineNet.

It is known that one major challenge for CNN-based methods lies in the difficulty to obtain a large number of labelled training data. For our case, there is no publicly available dataset that can provide large-scale face images with their corresponding high-quality 3D face models. For training CoarseNet, [41] and [42] resolve the training data problem by directly synthesizing face images with randomized parametric face model parameters. Nevertheless, due to the low dimensionality of the parametric face model, albedo and random background synthesized, the rendered images in [41], [42] are not photo-realistic. In contrast, we propose to create realistic face images by starting from real photographs and manipulating them after an inverse rendering procedure. For training FineNet, because of no dataset with detailed face geometry, [42] uses an unsupervised training by adopting the shading energy as the loss function. However, to make back-propagation trackable, [42] employs the first-order spherical harmonics to model the lighting, which makes the final detailed reconstruction not so accurate. On the contrary, we propose a novel approach to transfer different scales of details from one image to another. With the constructed fine-detailed face image dataset, we can train FineNet in a fully supervised manner, instead of the unsupervised way in [42], and thus can produce more accurate reconstruction results. Moreover, for training our coarse-scale tracking network for the video input case, we consider the coherence between adjacent frames and simulate adjacent frames according to the statistics learned from real facial videos for training data generation.

Contributions. In summary, the main contributions of this paper lie in the following five aspects:

- the optimization-based face inverse rendering that recovers accurate geometry, albedo, lighting from a single image, with which we can generate a large number of photo-realistic face images with different attributes to train our networks.
- a large photo-realistic face image dataset with the labels of the parametric face model parameters and the pose parameters, which are generated based on our proposed inverse rendering. This dataset facilitates the training of our Single-image CoarseNet and makes our method robust to expressions and poses.
- a large photo-realistic fine-scale face image dataset with detailed geometry labels, which are generated by our proposed face detail transfer approach. This fine-scale dataset facilitates the training of our FineNet.
- a large dataset for training Tracking CoarseNet, where we extend the Single-image CoarseNet training data by simulating their previous frames according to the statistics learned from real facial videos.
- the proposed 3DFaceNet that is trained with our built large-scale diverse synthetic data and is thus able to reconstruct the fine-scale geometry, albedo and lighting well in real time from monocular RGB video as well a single image. Our system is robust to large poses, extreme expressions and fast moving faces.

To the best of our knowledge, the proposed framework is the first work that achieves real-time dense 3D face reconstruction and tracking from monocular video. It might open up a new venue of research in the field of 3D assisted face video analysis. Moreover, the optimization-based face inverse rendering approach provides a novel, efficient way to generate various large-scale synthetic dataset by appropriate adaptation. Our elaborately-generated datasets will also benefit the face analysis related research that usually requires large amounts of training data.

2 Related Work

3D face reconstruction and facial performance capturing have been studied extensively in computer vision and computer graphics communities. For conciseness, we only review the most relevant works here.

Low-Dimensional Face Models. Model-based approaches for face shape reconstruction have grown in popularity over the last decade. Blanz and Vetter [4] proposed to represent a textured 3D face with principal components analysis (PCA), which provides an effective low-dimensional representation in terms of latent variables and corresponding basis vectors [50]. The model has been widely used in various computer vision tasks, such as face recognition [5], [54], face alignment [27], [34], [60], and face reenactment [53]. Although such a model is able to capture the global structure of a 3D face from a single image [4] or multiple images [2], the facial details like wrinkles and folds are not possible to be captured. In addition, the reconstructed face models rely heavily on training samples. For example, a face shape is difficult to be reconstructed if it is far away from the span of the training samples. Thus, similar to [42], we only use the low-dimensional model in our coarse layer to reconstruct a rough geometry and we refine the geometry in our fine layer.

Shape-From-Shading (SFS). SFS [39] makes use of the rendering principle to recover the underlying shape from shading observations. The performance of SFS largely depends on constraints or priors. For 3D face reconstruction, in order to achieve plausible results, the prior knowledge about the geometry must be applied. For instance, in order to reduce the ambiguity and the complexity of SFS, the symmetry of the human face has often been employed [49], [58], [59]. Kemelmacher et al. [29] used a reference model prior to align with the face image and then applied SFS to refine the reference model to better match the image. Despite the improved performance of this technique, its capability to capture global face structure is limited.

Inverse Rendering. The generation of a face image depends on several factors: face geometry, albedo, lighting, pose and camera parameters. Face inverse rendering refers to the process of estimating all these factors from a real face image, which can then be manipulated to render new images. Inverse rendering is similar to SFS with the difference that

inverse rendering aims to estimate all the rendering parameters while SFS mainly cares about reconstructing the geometry. Aldrian et al. [1] did face inverse rendering with a parametric face model using a multilinear approach, where the face geometry and the albedo are encoded on parametric face model. In [1], the geometry is first estimated based on the detected landmarks, and then the albedo and the lighting are iteratively estimated by solving the rendering equation. However, since the landmark constraint is a sparse constraint, the reconstructed geometry may not fit the face image well. [18] fits a 3D face in a multi-layer approach and extracts a high-fidelity parameterized 3D rig that contains a generative wrinkle formation model capturing the person-specific idiosyncrasies. [3] presents an algorithm for fully automatically fitting a 3D Morphable Model to a single image using landmarks and edge features. [46] introduces a framework to fit a parametric face model with Bayesian inference. [13] and [14] estimate an occlusion map and fit a statistical model to a face image with an EM-like probabilistic estimation process. [26] adopts the similar approach to recover the 3D face model with geometry details. While these methods provide impressive results, they are usually time-consuming due to complex optimization.

Face Capture from RGB Videos. Recently, a variety of methods have been proposed to do 3D face reconstruction with monocular RGB video. Most of them use a 3D Morphable Model [18], [22], [53] or a multi-linear face model [7], [8], [9], [45], [48] as a prior. [15] reconstructs the dense 3D face from a monocular video sequence by a variational approach, which is formulated as estimating dense low-rank smooth 3D shapes for each frame of the video sequence. [17] adapts a generic template to a static 3D scan of an actor's face, then fits the blendshape model to monocular video off-line, and finally extracts surface detail by shading-based shape refinement under general lighting. [48] uses a similar tracking approach and achieves impressive results based on global energy optimization of a set of selected keyframes. [18] fits a 3D face in a multi-layer approach and extracts a high-fidelity parameterized 3D rig that contains a generative wrinkle formation model capturing the person-specific idiosyncrasies. Although all these methods provide impressive results, they are time-consuming and are not suitable for real-time face video reconstruction and editing. [8], [9] adopt a learning-based regression model to fit a generic identity and expression model to a RGB face video in real-time and [7] extends this approach by also regressing fine-scale face wrinkles. [45] presents a method for unconstrained real-time 3D facial performance capture through explicit semantic segmentation in the RGB input. [22] tracks face by fitting 3D Morphable Model to the detected landmarks. Although they are able to reconstruct and track 3D face in real-time, they do not estimate facial appearance. Recently, [53] presented an approach for real-time face tracking and facial reenactment, but the method is not able to recover fine-scale details and requires external landmark inputs. In contrast, our method is the first work that can do real-time reconstruction of face geometry at fine details as well as real-time recovery of albedo, lighting and pose parameters.

Learning-Based Single-Image 3D Face Reconstruction. With the powerfulness of convolution neural networks, deep learning based methods have been proposed to do 3D face reconstruction from one single image. [27], [31], [60] use 3D Morphable Model (3DMM) [4] to represent 3D faces and use CNN to learn the 3DMM and pose parameters. [41] follows the method and uses synthetic face images generated by rendering textured 3D faces encoded on 3DMM with random lighting and pose for training data. However, the reconstruction results of these methods do not contain geometry details. Besides learning the 3DMM and pose parameters, [42] extends these methods by also learning detailed geometry in an unsupervised manner. [54] proposes to regress robust and discriminative 3DMM with a very deep neural network and uses it for face recognition. [51] proposes to use an analysis-by-synthesis energy function as the loss function during network training [4], [53]. [24] proposes to directly regress volumes with CNN for a single face image. Although these methods utilize the powerfulness of CNNs, they all concentrate on images and do not account for videos. In comparison, we focus on monocular face video input and reconstruct face video in real-time by using CNNs.

3 Face Rendering Process

This section describes some background information, particularly on the face representations and the face rendering process considered in our work. The rendering process of a face image depends on several factors: face geometry, albedo, lighting, pose and camera parameters. We encode 3D face geometry into two layers: a coarse-scale shape and fine-scale details. While the coarse-scale shape and albedo are represented by a parametric textured 3D face model, the fine-scale details are represented by a pixel depth displacement map. The face shape is represented via a mesh of n vertices with fixed connectivity as a vector $\mathbf{p} = [\mathbf{p}_1^T, \mathbf{p}_2^T, \ldots, \mathbf{p}_n^T]^T \in \mathcal{R}^{3n}$, where $\mathbf{p}_i$ denotes the position of vertex v_i $(i = 1, 2, \ldots, n)$.

Parametric Face Model. We use 3D Morphable Model [4] as the parametric face model to encode 3D face geometry and albedo on a lower-dimensional subspace, and extend the shape model to also cover facial expressions by adding delta blendshapes. Specifically, the parametric face model describes 3D face geometry $\mathbf{p}$ and albedo $\mathbf{b}$ with PCA (principle component analysis)

$$\mathbf{p} = \bar{\mathbf{p}} + \mathbf{A}_{\mathrm{id}}\boldsymbol{\alpha}_{\mathrm{id}} + \mathbf{A}_{\mathrm{exp}}\boldsymbol{\alpha}_{\mathrm{exp}}, \tag{1}$$

$$\mathbf{b} = \bar{\mathbf{b}} + \mathbf{A}_{\mathrm{alb}}\boldsymbol{\alpha}_{\mathrm{alb}}, \tag{2}$$

where $\bar{\mathbf{p}}$ and $\bar{\mathbf{b}}$ denote respectively the shape and the albedo of the average 3D face, $\mathbf{A}_{\mathrm{id}}$ and $\mathbf{A}_{\mathrm{alb}}$ are the principle axes extracted from a set of textured 3D meshes with a neutral expression, $\mathbf{A}_{\mathrm{exp}}$ represents the principle axes trained on the offsets between the expression meshes and the neutral meshes of individual persons, and $\boldsymbol{\alpha}_{\mathrm{id}}$, $\boldsymbol{\alpha}_{\mathrm{exp}}$ and $\boldsymbol{\alpha}_{\mathrm{alb}}$ are the corresponding coefficient vectors that characterize a specific 3D face model. For diversity and mutual complement, we use the Basel Face Model (BFM) [37] for $\mathbf{A}_{\mathrm{id}}$ and $\mathbf{A}_{\mathrm{alb}}$ and FaceWarehouse [10] for $\mathbf{A}_{\mathrm{exp}}$.

Fine-Scale Details. As 3DMM is a low-dimensional model, some face details such as wrinkles and dimples cannot be expressed by 3DMM. Thus, we encode the geometry details in a displacement along the depth direction for each pixel.

Rendering Process. For camera parametrization, following [42], we use the weak perspective model to project the 3D face onto the image plane

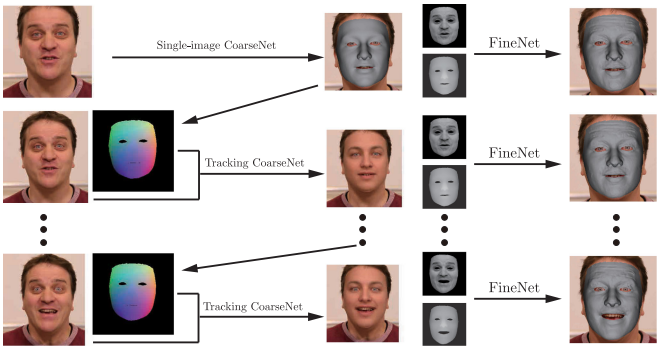


Fig. 1. The pipeline of our proposed learning based dense 3D face reconstruction and tracking framework. The first frame of the input video is initially reconstructed by a single-image CoarseNet for coarse face geometry reconstruction, followed by using FineNet for detailed face geometry recovery. Each of the subsequent frames is processed by a tracking CoarseNet followed by FineNet.

$$\mathbf{q}_i = s\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{pmatrix}R\mathbf{p}_i + \mathbf{t}, \tag{3}$$

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, s is the scale factor, R is the rotation matrix constructed from Euler angles $pitch, yaw, roll$ and $\mathbf{t} = (t_x, t_y)^T$ is the translation vector.

To model the scene lighting, we assume the face to be a Lambertian surface. The global illumination is approximated using the spherical harmonics (SH) basis functions [35]. Then, the irradiance of a vertex v_i with surface normal $\mathbf{n}_i$ and scalar albedo b_i is expressed as [40]

$$\mathcal{L}(\mathbf{n}_i, b_i \mid \boldsymbol{\gamma}) = b_i \cdot \sum_{k=1}^{B^2} \gamma_k \phi_k(\mathbf{n}_i), \tag{4}$$

where $\boldsymbol{\phi}(\mathbf{n}_i) = [\phi_1(\mathbf{n}_i), \ldots, \phi_{B^2}(\mathbf{n}_i)]^T$ is the SH basis functions computed with normal $\mathbf{n}_i$, and $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_{B^2}]^T$ is the SH coefficients. We use the first $B = 3$ bands of SHs for the illumination model. Thus, the rendering process depends on the parameter set $\chi = \{\boldsymbol{\alpha}_{\text{id}}, \boldsymbol{\alpha}_{\text{exp}}, \boldsymbol{\alpha}_{\text{alb}}, s, pitch, yaw, roll, \mathbf{t}, \mathbf{r}\}$, where $\mathbf{r} = (\boldsymbol{\gamma}_r^T, \boldsymbol{\gamma}_g^T, \boldsymbol{\gamma}_b^T)^T$ denotes RGB channels' SH illumination coefficients.

Given the parametric face model and the parameter set χ, a face image can be rendered as follows. First, a textured 3D mesh is constructed using Eqs. (1) and (2). Then we do a rasterization via Eq. (3). Particularly, in the rasterization, for every pixel in the face region of the 2D image, we obtain the underlying triangle index on the 3D mesh and its barycentric coordinates. In this way, for every pixel in the face region, we obtain its normal by using the underlying triangle's normal, and its albedo value by barycentrically interpolating the albedos of the vertices of the underlying triangle. Finally, with the normal, the albedo and the lighting, the color of a pixel can be rendered using Eq. (4).

4 Overview of Proposed Learning-Based Dense Face Reconstruction

To achieve real-time face video reconstruction and tracking, we need real-time face inverse rendering. However, reconstructing detailed 3D face using traditional optimization-based methods [18] is far from real-time. To address this problem, we develop a novel CNN based framework to achieve real-time detailed face inverse rendering. Specifically, we use two CNNs for each frame, namely CoarseNet and FineNet. The first one estimates coarse-scale geometry, albedo, lighting and pose parameters altogether, and the second one reconstructs the fine-scale geometry encoded on pixel level.

Fig. 1 shows the entire system pipeline. It can be seen that there are two types of CoarseNet: Single-image CoarseNet and Tracking CoarseNet. Tracking CoarseNet makes use of the predicted parameters of the previous frame, while Single-image CoarseNet is for the first frame case where there is no previous frame available. Such Single-image CoarseNet could be applied to other key frames as well to avoid any potential drifting problem if needed. The combination of all the networks including Single-image CoarseNet, Tracking CoarseNet and FineNet, makes up a complete framework for real-time dense 3D face reconstruction from monocular video. Note that the entire framework can be easily degenerated to the solution for dense 3D face reconstruction from a single image by combining only Single-image CoarseNet with FineNet.

We would like to point out that although we advocate the CNN based solution, it still needs to work together with optimization based inverse rendering methods. This is because CNN requires large amount of data with labels, which is usually not available, and optimization based inverse rendering methods are a natural solution for generating labels (optimal parameters) and synthesizing new images offline. Thus, our proposed dense face reconstruction and tracking framework includes both optimization based inverse rendering and the two-stage CNN based solution, where the former is for offline training data generation and the latter is for real-time online operations. In the subsequent sections, we first introduce our optimization based inverse face rendering, which will be used to construct training data for CoarseNet and FineNet; and then we present our three convolutional networks.

5 Optimization Based Face Inverse Rendering

Inverse rendering is an inverse process of image generation. That is, given a face image, we want to estimate a 3D face with albedo, lighting condition, pose and projection parameters simultaneously. Since directly estimating these unknowns with only one input image is an ill-posed problem, we use the parametric face model as a prior. Fig. 2 illustrates our developed inverse rendering, which consists of three stages: parametric face model fitting, geometry refinement and albedo blending. The first stage is to recover the lighting, a coarse geometry and the albedo based on the parametric face model. The second stage is to further recover the geometry details. The third stage is to blend the albedo so as to make the rendered image closer to the input image. Via the developed inverse rendering, we are able to extract different rendering components of real face images, and then by varying these different components we can create large-scale photo-realistic face images to facilitate the subsequent CNN based training.

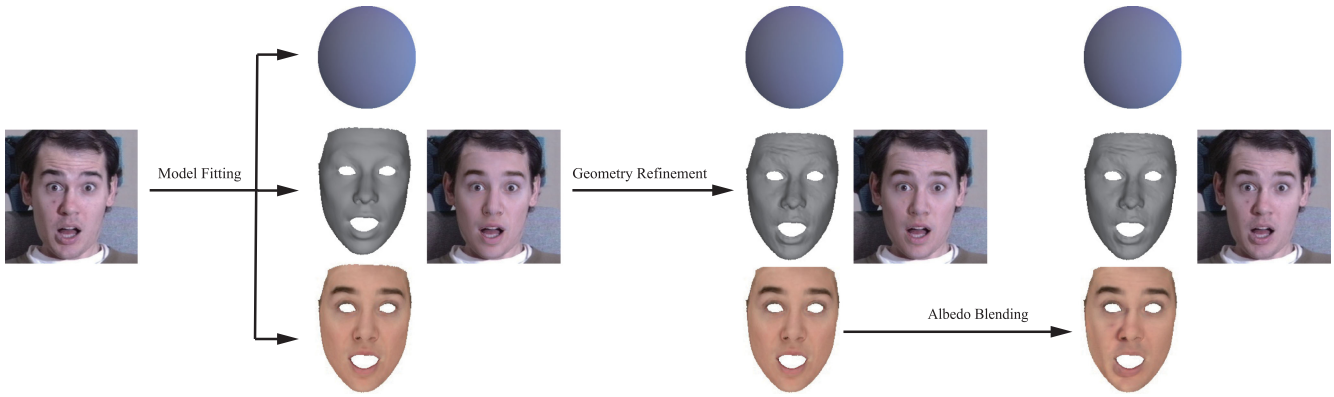


Fig. 2. The pipeline of our proposed inverse rendering method. Given an input face image (left), our inverse rendering consists of three stages: Model fitting (second column), geometry refinement (third column) and albedo blending (last column). At each stage, the top to bottom rows are the corresponding recovered lighting, geometry and albedo, and the rendered face image is shown on the right. The arrows indicate which component is updated.

5.1 Stage 1-Model Fitting

The purpose of model fitting is to estimate the coarse face geometry, albedo, lighting, pose and projection parameters from a face image I_{in}. That is to estimate $\chi = \{\boldsymbol{\alpha}_{\text{id}}, \boldsymbol{\alpha}_{\text{exp}}, \boldsymbol{\alpha}_{\text{alb}}, s, pitch, yaw, roll, \mathbf{t}, \mathbf{r}\}$. For convenience, we group these parameters into the following sets $\chi_g = \{\boldsymbol{\alpha}_{\text{id}}, \boldsymbol{\alpha}_{\text{exp}}\}$, $\chi_p = \{pitch, yaw, roll\}$, $\chi_t = \{s, \mathbf{t}\}$ and $\chi_l = \{\boldsymbol{\alpha}_{\text{alb}}, \mathbf{r}\}$. The fitting process is based on the analysis-by-synthesis strategy [4], [53], and we seek a solution that by minimizes the difference between the input face image and the rendered image with χ. Specifically, we minimize the following objective function

$$E(\chi) = E_{\text{con}} + w_l E_{\text{lan}} + w_r E_{\text{reg}}, \tag{5}$$

where E_{con} is a photo-consistency term, E_{lan} is a landmark term and E_{reg} is a regularization term, and w_l and w_r are tradeoff parameters. The photo-consistency term, aiming to minimize the difference between the input face image and the rendered image, is defined as

$$E_{\text{con}}(\chi) = \frac{1}{|\mathcal{F}|}\|I_{\text{ren}} - I_{\text{in}}\|^2, \tag{6}$$

where I_{ren} is the rendered image, I_{in} is the input image, and $\mathcal{F}$ is the set of all pixels in the face region. The landmark term aims to make the projected vertices close to the corresponding landmarks in the image plane

$$E_{\text{lan}}(\chi) = \frac{1}{|\mathcal{L}|}\sum_{i\in\mathcal{L}}\|\mathbf{q}_i - (\Pi R\mathbf{p}_i + \mathbf{t})\|^2, \tag{7}$$

where $\mathcal{L}$ is the set of landmarks, $\mathbf{q}_i$ is a landmark position in the image plane, $\mathbf{p}_i$ is the corresponding vertex location in the fitted 3D face and $\Pi = s\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{pmatrix}$. The regularization term aims to ensure that the fitted parametric face model parameters are plausible

$$E_{\text{reg}}(\chi) = \sum_{i=1}^{100}\left[\left(\frac{\boldsymbol{\alpha}_{\text{id},i}}{\boldsymbol{\sigma}_{\text{id},i}}\right)^2 + \left(\frac{\boldsymbol{\alpha}_{\text{alb},i}}{\boldsymbol{\sigma}_{\text{alb},i}}\right)^2\right] + \sum_{i=1}^{79}\left(\frac{\boldsymbol{\alpha}_{\text{exp},i}}{\boldsymbol{\sigma}_{\text{exp},i}}\right)^2, \tag{8}$$

where $\boldsymbol{\sigma}$ is the standard deviation of the corresponding principal direction. Here we use 100 principle components for identity & albedo, and 79 for expression. In our experiments, we set w_l to be 10 and w_r to be $5 \cdot 10^{-5}$. Eq. (5) is minimized via Gauss-Newton iteration.

5.2 Stage 2-Geometry Refinement

As the parametric face model is a low-dimensional model, some face details such as wrinkles and dimples are not encoded in parametric face model. Thus, the purpose of the second stage is to refine the geometry by adding the geometry details in a displacement along the depth direction for every pixel. In particular, by projecting the fitted 3D face with parameter χ, we can obtain a depth value for every pixel in the face region. Let $\mathbf{z}$ be all stacked depth values of pixels, $\mathbf{d}$ be all stacked displacements and $\widetilde{\mathbf{z}} = \mathbf{z} + \mathbf{d}$ be all new depth values. Given new depth values $\widetilde{\mathbf{z}}$, the normal at pixel (i, j) can be computed using the normal of triangle $(\mathbf{p}_{(i,j)}, \mathbf{p}_{(i+1,j)}, \mathbf{p}_{(i,j+1)})$, where $\mathbf{p}_{(i,j)} = [i, j, \widetilde{\mathbf{z}}(i, j)]^T$ is the coordinates of pixel (i, j) at the camera system. Inspired by [23], we estimate $\mathbf{d}$ using the following objective function:

$$E(\mathbf{d}) = E_{\text{con}} + \mu_1\|\mathbf{d}\|_2^2 + \mu_2\|\triangle\mathbf{d}\|_1, \tag{9}$$

where E_{con} is the same as that in Eq. (5), $\|\mathbf{d}\|_2^2$ is to encourage small displacements, the Laplacian of displacements $\triangle\mathbf{d}$ is to make the displacement smooth, and μ_1 and μ_2 are tradeoff parameters. We use ℓ_1 norm for the smooth term as it allows preserving sharp discontinuities while removing noise. We set μ_1 to be $1 \cdot 10^{-3}$ and μ_2 to be 0.3 in our experiments. Eq. (9) is minimized by using an iterative reweighing approach [11].

5.3 Stage 3-Albedo Blending

Similar to the geometry, the albedo encoded in the parametric face model (denoted as $\mathbf{b}_c$) in stage 1 is also smooth because of the low dimension. For photo-realistic rendering, we extract a fine-scale albedo as

$$\mathbf{b}_f = I_{\text{in}}./(\mathbf{r}^T\boldsymbol{\phi}(\mathbf{n})), \tag{10}$$

where $./$ represents the elementwise division operation, I_{in} is the color of the input image and $\mathbf{n}$ is the normal computed from the refined geometry. However, the fine-scale albedo $\mathbf{b}_f$ might contain some geometry details due to imperfect geometry refinement. To avoid this, we linearly blend $\mathbf{b}_c$ and $\mathbf{b}_f$, i.e., $\beta\mathbf{b}_c + (1-\beta)\mathbf{b}_f$, with different weights β at different regions. Particularly, in the regions where geometry details are likely to appear such as forehead and eye corners, we make the blended albedo close to $\mathbf{b}_c$ by setting β to

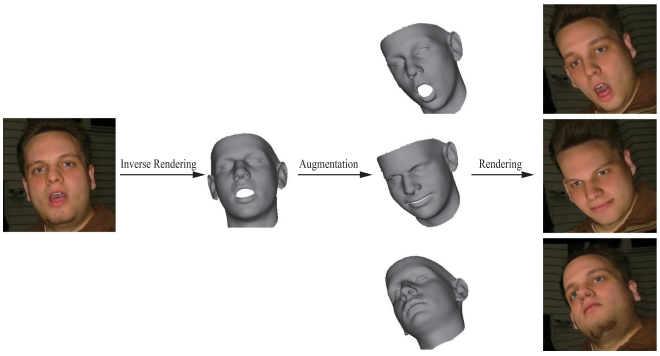


Fig. 3. Training data synthesis for Single-image CoarseNet. Given a real face image, we first do the inverse rendering to estimate lighting, albedo and geometry. Then, by changing the expression parameter α_{exp} and the pose parameters $pitch, yaw, roll$, the face geometry is augmented. In the final, a set of new face images is obtained by rendering the newly changed face geometry.

be 0.65, while in the other regions we encourage the blended albedo close to $\mathbf{b}_f$ by setting β to be 0.35. Around the border of the regions β is set continuously from 0.35 to 0.65. Finally, we use this blended albedo as $\mathbf{b}$ in Eq. (4) for our subsequent data generation process.

6 Single-Image CoarseNet for Coarse Reconstruction from a Single Image

In this section, we describe how to train a coarse-layer CNN (called Single-image CoarseNet) that can output the parametric face model parameters (corresponding to a coarse shape) and the pose parameters from the input of a single face image or an independent video frame. Although the network structure of Single-image CoarseNet is similar to that of [42], [60], we use our uniquely constructed training data and loss function, which are elaborated below.

6.1 Constructing Single-Image CoarseNet Training Data

To train Single-image CoarseNet, we need a large-scale dataset of face images with ground-truth 3DMM parameters and pose parameters. Recently, [60] proposed to synthesize a large number of face images by varying the 3DMM parameters fitted from a small number of real face images. [60] focuses on the face alignment problem. The color of the synthesized face images are directly copied from the source images without considering the underlying rendering process, which makes the synthesized images not photo-realistic and thus unsuitable for high-quality 3D face reconstruction. Later, [42] follows the idea of using synthetic data for learning detailed 3D face reconstruction and directly renders a large number of face images by varying the existing 3DMM parameters with random texture, lighting, and reflectance. However, since 3DMM is a low-dimensional model and the albedo is also of low frequency, the synthetic images in [42] are not photo-realistic as well, not to mention the random background used in the rendered images. In addition, the synthetic images in [42] are not available to the public.

Therefore, in this paper, we propose to use our developed inverse rendering described in Section 5 to synthesize photo-realistic images at large scale, which well addresses the shortcoming of the synthetic face images generated in [42], [60]. In particular, we choose 4,000 face images (dataset A), in which faces are not occluded, from 300 W [44] and Multi-pie [19]. For each of the 4,000 images, we use our optimization based inverse rendering method to obtain the parameter set χ. Then, to make our coarse-layer network robust to expression and pose, we render new face images by randomly changing the pose parameters χ_p and the expression parameter $\boldsymbol{\alpha}_{\text{exp}}$, each of which leads to a new parameter set $\widetilde{\chi}$. By doing the rasterization with $\widetilde{\chi}$, we can obtain the normals of all pixels in the new face region as described in Section 3. With these normals and the albedos obtained according to Section 5.3, a new face is then rendered using Eq. (4). We also warp the background region of the source image to fit the new face region by using the image meshing [60]. Fig. 3 shows an example of generating three synthetic images from an input real images by simultaneously changing the expression and pose parameters. In this way, we generate a synthetic dataset of totally 80,000 face images for the Single-image CoarseNet training by randomly varying the expression and the pose parameters 20 times for each of the 4,000 real face images.

6.2 Single-Image CoarseNet

The input to our Single-image CoarseNet is a face image, and the output is the parameters related to the shape of 3D face and the projection, i.e., $\mathcal{T} = \{\alpha_{\text{id}}, \alpha_{\text{exp}}, s, pitch, yaw, roll, t_x, t_y\}$. The network is based on the Resnet-18 [20] with the modification of changing the output number of the fully-connected layer to 185 (100 for identity, 79 for expression, 3 for rotation, 2 for translation and 1 for scale). The input image size is 224×224.

As pointed out in [60], different parameters in $\mathcal{T}$ have different influence to the estimated geometry. Direct mean square error (MSE) loss on $\mathcal{T}$ might not lead to good geometry reconstruction. [60] uses a weighted MSE loss, where the weights are based on the projected vertex distances. [42] uses 3D vertex distances to measure the loss from the geometry parameters and MSE for the pose parameters. Considering these vertex based distance measures are calculated on the vertex grid, which might not well measure how the parameters fit the input face image, in this work we use a loss function that computes the distance between the ground-truth parameters $\mathcal{T}_g$ and the network output parameters $\mathcal{T}_n$ at the per-pixel level.

In particular, we first do the rasterization with the ground-truth parameters $\mathcal{T}_g$ to get the underlying triangle index and the barycentric coordinates for each pixel in the face region. With this information, we then construct the pixels' 3D average $\bar{\mathbf{p}}_q$, base $\mathbf{A}_{q,\text{id}}$ and base $\mathbf{A}_{q,\text{exp}}$ by barycentrically interpolating the corresponding rows in $\bar{\mathbf{p}}$, $\mathbf{A}_{\text{id}}$, $\mathbf{A}_{\text{exp}}$, respectively. In this way, given parameters $\mathcal{T}$, we can project all the corresponding 3D locations of the pixels onto the image plane using

$$Proj(\mathcal{T}) = \Pi R(\bar{\mathbf{p}}_q + \mathbf{A}_{q,\text{id}}\boldsymbol{\alpha}_{\text{id}} + \mathbf{A}_{q,\text{exp}}\boldsymbol{\alpha}_{\text{exp}}) + \mathbf{t}. \quad (11)$$

Then the loss between the ground-truth parameters $\mathcal{T}_g$ and the network output parameters $\mathcal{T}_n$ is defined as

$$\mathcal{D}(\mathcal{T}_g, \mathcal{T}_n) = \|Proj(\mathcal{T}_g) - Proj(\mathcal{T}_n)\|_2^2. \quad (12)$$

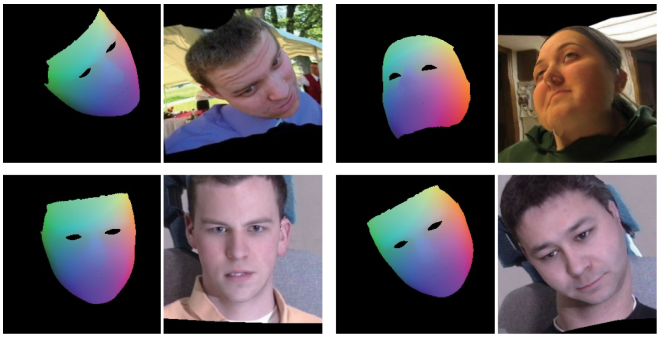


Fig. 4. Examples of adjacent frame simulations. For each pair, the left is the PNCC image generated by simulating the previous frame, and the right is the current face frame.

Note that there is no need to compute $Proj(\mathcal{T}_g)$ since it corresponds to the original pixel locations in the image plane.

For better convergence, we further separate the loss in Eq. (12) into the pose-dependent loss as

$$\mathcal{L}_{\text{pose}} = \|Proj(\mathcal{T}_g) - Proj(\mathcal{T}_{n,\text{pose}}, \mathcal{T}_{g,\text{geo}})\|_2^2, \tag{13}$$

where $\mathcal{T}_{\text{pose}} = \chi_p \cup \chi_t$ represents the pose parameters, and the geometry-dependent loss as

$$\mathcal{L}_{\text{geo}} = \|Proj(\mathcal{T}_g) - Proj(\mathcal{T}_{n,\text{geo}}, \mathcal{T}_{g,\text{pose}})\|_2^2, \tag{14}$$

where $\mathcal{T}_{\text{geo}} = \chi_g$ represents the geometry parameters. In Eqs. (13) and (14), $Proj(\mathcal{T}_{n,\text{pose}}, \mathcal{T}_{g,\text{geo}})$ (resp., $Proj(\mathcal{T}_{n,\text{geo}}, \mathcal{T}_{g,\text{pose}})$) refers to the projection with the ground-truth geometry (resp., pose) parameters and the network estimated pose (resp., geometry) parameters.

The final loss is a weighted sum of the two losses

$$\mathcal{L} = w \cdot \mathcal{L}_{\text{pose}} + (1 - w) \cdot \mathcal{L}_{\text{geo}}, \tag{15}$$

where w is the tradeoff parameter. We set $w = \frac{\mathcal{L}_{\text{geo}}}{\mathcal{L}_{\text{pose}} + \mathcal{L}_{\text{geo}}}$ for balancing the two losses and we assume w is a constant when computing the derivatives for back propagation.

7 Tracking CoarseNet for Coarse Reconstruction from Monocular Video

The purpose of Tracking CoarseNet is to predict the current frame's parameters, given not only the current video frame but also the previous frame's parameters. As there does not exist large-scale dataset that captures the correlations among adjacent video frames, our Tracking CoarseNet also faces the problem of no sufficient well-labelled training data. Similarly, we synthesize training data for Tracking CoarseNet. However, it is non-trivial to reuse the $(k-1)$th frame's parameters to predict kth frame's parameters. Directly using all the previous frame's parameters as the input to Tracking CoarseNet will introduce too many uncertainties during training, which results in huge complexity in synthesizing adjacent video frames for training, and make the training hard to converge and the testing unstable. Through vast experiments, we find that only utilizing the previous frame's pose parameters is a good way to inherit the coherence while keeping the network trainable and stable.

Specifically, the input to the tracking network is the kth face frame cropped by the $k-1$ frame's landmarks and a Projected Normalized Coordinate Code (PNCC) [60] rendered using the $k-1$ frame's pose parameters χ_p^{k-1}, χ_t^{k-1} and the mean 3D face $\bar{\mathbf{p}}$ in Eq. (1). The output of the tracking network is parameters $\mathcal{T}^k = \{\boldsymbol{\alpha}_{\text{id}}^k, \boldsymbol{\alpha}_{\text{exp}}^k, \boldsymbol{\alpha}_{\text{alb}}^k, \delta^k(s), \delta^k(pitch), \delta^k(yaw), \delta^k(roll), \delta^k(\mathbf{t}), \mathbf{r}^k\}$, where $\delta(\cdot)$ denotes the difference between the current frame and the previous frame. Note that here the output also includes albedo and lighting parameters, which could be used for different video editing applications.

The network structure is the same as Single-image CoarseNet except that the output number of the fully-connected layer is 312 (100 for identity, 79 for expression, 3 for rotation, 2 for translation, 1 for scale, 100 for albedo and 27 for lighting coefficients). In addition to the loss terms $\mathcal{L}_{\text{pose}}$ and $\mathcal{L}_{\text{geo}}$ defined in Eqs. (13) and (14) respectively, Tracking CoarseNet also uses another term for $\boldsymbol{\alpha}_{\text{alb}}^k$ and $\mathbf{r}^k$ that measures the distance between the rendered image and the input frame

$$\mathcal{L}_{\text{col}} = \|I_{\text{ren}}^k(\boldsymbol{\alpha}_{\text{alb}}^k, \mathbf{r}^k) - I_{\text{in}}^k\|_2^2, \tag{16}$$

where $I_{\text{ren}}^k(\boldsymbol{\alpha}_{\text{alb}}^k, \mathbf{r}^k)$ is the rendered face image with the groundtruth geometry and pose, and the estimated albedo and lighting, and I_{in}^k is the input face frame. In this way, the final total loss becomes a weighted sum of the three losses

$$\mathcal{L} = w_1 \cdot \mathcal{L}_{\text{pose}} + w_2 \cdot \mathcal{L}_{\text{geo}} + (1 - w_1 - w_2) \cdot \mathcal{L}_{\text{col}}, \tag{17}$$

where $w_1 = \frac{\mathcal{L}_{\text{geo}} + \mathcal{L}_{\text{col}}}{2(\mathcal{L}_{\text{pose}} + \mathcal{L}_{\text{geo}} + \mathcal{L}_{\text{col}})}$ and $w_2 = \frac{\mathcal{L}_{\text{pos}} + \mathcal{L}_{\text{col}}}{2(\mathcal{L}_{\text{pose}} + \mathcal{L}_{\text{geo}} + \mathcal{L}_{\text{col}})}$ are the tradeoff parameters to balance the three losses, and we assume w_1 and w_2 are constant when computing the derivatives for back propagation.

Training Data Generation for Tracking CoarseNet. To train Tracking CoarseNet, large-scale adjacent video frame pairs with ground-truth parameters χ are needed as training data. Again, there is no such public dataset. To address this problem, we propose to simulate adjacent video frames, i.e., to generate the previous frame for each of the 80,000 synthesized images used in the Single-image CoarseNet training. Randomly varying the parameter set $\widetilde{\chi}$ for a training image does not capture the tight correlations among adjacent frames. Thus, we propose to do simulation by analysing the distribution of the previous frame's parameters χ^{k-1} given the current kth frame from real videos. Considering our tracking network only makes use of the previous frame's pose parameters, we just need to obtain the distribution of χ_p^{k-1} and χ_t^{k-1} given χ_p^k and χ_t^k. Particularly, we assume each parameter in $\delta^k(\chi_p) = \chi_p^{k-1} - \chi_p^k$ and $\delta^k(\chi_t) = \chi_t^{k-1} - \chi_t^k$ follows normal distribution. We extract about 160,000 adjacent frame pairs from the 300-VW video dataset [47] and use our Single-image CoarseNet to get the parameters for fitting the normal distribution. Finally, for each of the 80,000 synthesized images, we can simulate its previous frame by generating $\widetilde{\chi}_p^{k-1}$ and $\widetilde{\chi}_t^{k-1}$ according to the obtained normal distribution. Examples of several simulated pairs with the previous frame's PNCC and the current image are shown in Fig. 4.

8 FineNet for Fine-Scale Geometry Reconstruction

In this section, we present our solution on how to train a fine-layer CNN (called FineNet). The input to FineNet is a coarse

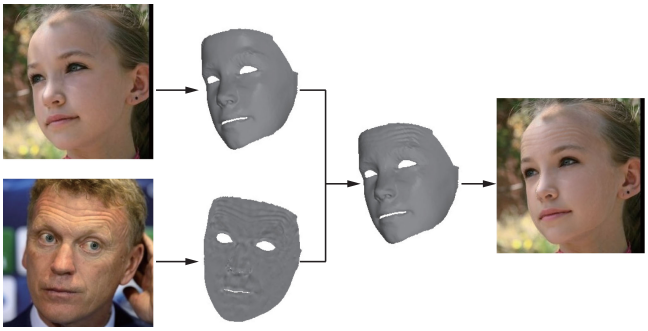


Fig. 5. Synthetic data generation for training FineNet. Given a target face image without many geometry details (top left) and a source face image (bottom left) that is rich of wrinkles, we first apply our developed inverse rendering on both images to obtain the projected geometry for target face (top second) and a displacement map for the source face (bottom right). Then we transfer the displacement map of the source face to the geometry of the target face. Finally we render the updated geometry to get a new face image (top right) which contains the same type of wrinkles as the source face.

depth map stacked with the face image. The coarse depth map is generated by using the method described in Section 5.2 with the parameters $\mathcal{T}$ estimated by either Single-image CoarseNet or Tracking CoarseNet. The output of our FineNet is a per-pixel displacement map. Again, the key challenge here is that there is no fine-scale face dataset available that can provide a large number of detailed face geometries with their corresponding 2D images, as pointed out in [42]. In addition, the existing morphable face models such as 3DMM cannot capture the fine-scale face details. [42] bypasses this challenge by converting the problem into an unsupervised setting, i.e., relating the output depth map to the 2D image by using the shading energy as the loss function. However, to make the back-propagation trackable under the shading energy, they have to use first-order spherical harmonics to model the lighting, which is not accurate.

In our work, instead of doing unsupervised training [42], we go for fully supervised training of FineNet, i.e., directly constructing a large-scale detailed face dataset based on our developed inverse rendering and a novel face detail transfer approach, which will be elaborated below. Note that our FineNet architecture is based on the U-Net [43] and we use Euclidean distance as the loss function.

8.1 Constructing FineNet Training Data

Our synthesized training data for FineNet is generated by transferring the displacement map from a source face image with fine-scale details such as wrinkles and folds to other target face images without the details. Fig. 5 gives such an example. In particular, we first apply our developed inverse rendering in Section 5 on both images. Then we find correspondences between the source image pixels and the target image pixels using the rasterization information described in Section 3. That is, for a pixel (i, j) in the target face region, if its underlying triangle is visible in the source image, we find its corresponding 3D location on the target 3D mesh by barycentric interpolation, and then we project the 3D location onto the source image plane using Eq. (3) to get the corresponding pixel (i', j'). With these correspondences, the original source displacement $\boldsymbol{d}_s$ and the original target displacement $\boldsymbol{d}_t$, a new displacement $\widetilde{\boldsymbol{d}}_t$ for the target image is generated by matching its gradients with the scaled source displacement gradient in the intersected region Ω by solving the following poisson problem

$$\begin{aligned} \min_{\widetilde{\mathbf{d}}_t} \quad & \sum_{(i,j)\in\Omega} \|\nabla\widetilde{\mathbf{d}}_t(i,j) - \mathbf{w}(i,j)\|^2, \\ \text{s.t.} \quad & \widetilde{\mathbf{d}}_t(i,j) = \mathbf{d}_t(i,j) \qquad (i,j) \in \partial\Omega, \end{aligned} \tag{18}$$

where $\mathbf{w}(i,j) = s_d[\mathbf{d}_s(i'+1,j') - \mathbf{d}_s(i',j'), \mathbf{d}_s(i',j'+1) - \mathbf{d}_s(i',j')]^T$ and s_d is a scale factor within the range $[0.7, 1.3]$ so as to create different displacement fields. After that, we add $\widetilde{\mathbf{d}}_t$ into the coarse target depth $\mathbf{z}$ to get the final depth map. Then the normals of the target face pixels are updated as in Section 5.2. With the updated normals, a new face image is rendered using Eq. (4).

We would like to point out that besides generating a large number of detailed face images to train the network, there are also other benefits to do such detail transfer. First, by rendering the same type of detail information under different lighting conditions, we can train our FineNet to be robust to lighting. Second, by changing the scale of the displacement randomly, our method can be trained to be robust to different scales of details.

For the details of the dataset construction, we first download 1,000 real face images (dataset B) that contain rich geometry details from internet. Then, we transfer the details from dataset B to the 4,000 real face images in dataset A, the one used in constructing synthetic data for Single-image CoarseNet. For every image in A, we randomly choose 30 images in B for transferring. In this way, we construct a synthesized fine-detailed face image dataset of totally 120,000 images.

9 Experiments

In this section, we conduct qualitative and quantitative evaluation on the proposed detailed 3D face reconstruction and tracking framework and compare it with the state-of-the-art methods.

Experimental Setup and Runtime. We train the CNNs via the CAFFE [25] framework. Single-image CoarseNet takes the input of a color face image with size $224 \times 224 \times 3$, and Tracking CoarseNet and FineNet respectively take the inputs of $256 \times 256 \times 6$ (a color image and a PNCC) and $256 \times 256 \times 2$ (a gray image and its coarse depth). We train all the networks using Adam solver with the mini-batch size of 100 and 30k iterations. The base learning rate is set to be 0.00005.

The CNN based 3D face reconstruction and tracking are implemented in C++ and tested on various face images and videos. All experiments were conducted on a desktop PC with a quad-core Intel CPU i7, 4 GB RAM and NVIDIA GTX 1,070 GPU. As for the running time for each frame, it takes 5 ms for CoarseNet and 15 ms for FineNet.

9.1 Results of Dense 3D Face Reconstruction from Monocular Video

CoarseNet versus FineNet. Our approach is to progressively and continuously estimate the detailed facial geometry, albedo and lighting parameters from a monocular face video. Fig. 6 shows the tracking output results of the two stages. The results of CoarseNet include the smooth geometry and the corresponding rendered face image shown in

Fig. 6. Results of our two-stage CNN based face tracking. Left: four frames of a video. Middle: results of Tracking CoarseNet (projected mesh and rendered face). Right: results of FineNet.

the middle column. The FineNet further predicts the pixel level displacement given in the last column. We can see that CoarseNet produces smooth geometry and well matched rendered face images, which show the good recovery of pose, albedo, lighting and projection parameters, and FineNet nicely recovers the geometry details such as wrinkles. A complete reconstruction results of all the video frames are given in the accompanying video or via the link: https://youtu.be/dghlMXxD-rk.

Single-Image CoarseNet versus Tracking CoarseNet. Given a RGB video, a straightforward way for dense face tracking is to treat all frames as independent face images, and apply our Single-image CoarseNet on each frame, followed by applying FineNet. Thus, we give a comparison of our proposed Tracking CoarseNet, which estimates the differences of the pose parameters w.r.t. the previous frame, with the baseline that simply uses our Single-image CoarseNet on each frame. As demonstrated in Fig. 7, Tracking CoarseNet

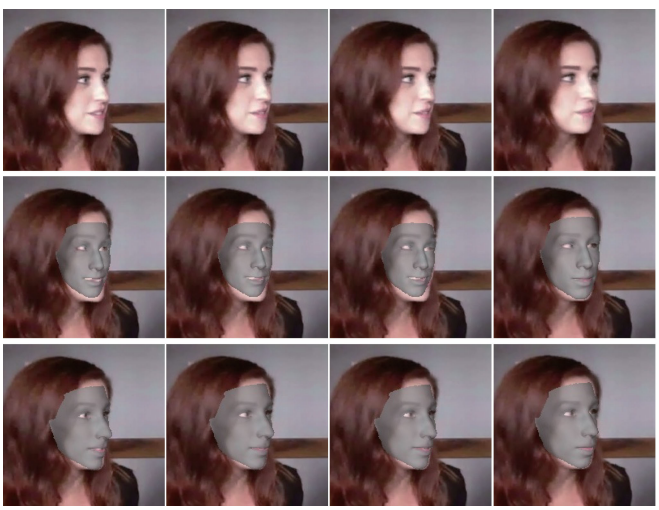


Fig. 7. Comparisons with image-based dense face tracking. Top row: four continuous frames from a video. Middle row: results of using our Single-image CoarseNet on each frame. Bottom row: results of our Tracking CoarseNet. It can be observed that Tracking CoarseNet achieves more robust tracking.

Fig. 8. Comparisons with the state-of-art dense face tracking methods [18], [22], [48]. The average computation time for each frame is given in the bracket. [48] does not report the running time of their method, while it should take much longer time than ours since it iteratively solves several complex optimization problems.

achieves more robust tracking than the baseline, since it well utilizes the guidance from the previous frame's pose.

Comparisons with Dense Face Tracking Methods. We compare our method with the state-of-the-art monocular video based dense face tracking methods [18], [22], [48]. [48] performs 3D face reconstruction in an iterative manner. In each iteration, they first reconstruct coarse-scale facial geometry from sparse facial features and then refine the geometry via shape from shading. [18] employs a multi-layer approach to reconstruct fine-scale details. They encode different scales of 3D face geometry on three different layers and do optimization for each layer. [22] reconstructs the 3D face shape by only fitting the 2D landmarks via 3DMM, and we can observe that [22] can only produce smooth face reconstruction. As shown in Fig. 8, our method produces visually better results compared to [22], and comparable results compared to [18] and [48].

Different from optimization based methods, our learning based approach is much faster while obtaining comparable or better results. Our method is several orders of magnitude faster than the state-of-the-art optimization-based approach [18], i.e., 5 ms for CoarseNet and 15 ms for FineNet with our hardware setting, while 175.5s reported in their paper [18]. It needs to be pointed out that the existing optimization based dense tracking methods need facial landmark constraints. Therefore, they might not reconstruct well for faces with large poses and extreme expressions. On the other hand, we do large-scale photo-realistic image synthesis that includes many challenging data with well labelled parameters, and thus we can handle those challenging cases as demonstrated in Fig. 9.

Quantitative Results of Face Reconstruction from Monocular Video. For quantitative evaluation, we test on the FaceCap dataset [55]. The dataset consists of 200 frames along with 3D meshes constructed using the binocular approach. We compare our proposed inverse rendering approach and our learning based solutions including Tracking CoarseNet and

Fig. 9. Reconstruction results for faces with large poses and extreme expressions. Top row: several frames from one input video. Bottom row: the reconstructed face shapes with geometry details. See the complete sequence in the accompanying video or via the link: https://youtu.be/dghlMXxD-rk.

TABLE 1
Quantitative Results of Dense Face Reconstruction from Monocular Video

Average point-to-point distance (mm)		
Inverse rendering	CoarseNet	CoarseNet+FineNet
1.81	2.11	2.08

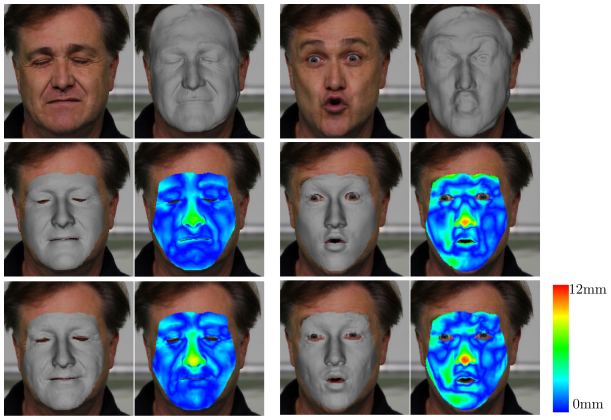


Fig. 10. Comparisons of our inverse rendering and our learning based dense face tracking solution. From top to bottom: input face video frame and groundtruth mesh in dataset [55], results of the inverse rendering approach, results of our learning based dense face tracking solution.

Tracking CoarseNet+FineNet. For each method, we register the depth cloud to the groundtruth 3D mesh and compare point to point distance. Table 1 shows the average point-to-point distance results. It can be seen that our proposed inverse rendering achieves an average distance of 1.81 mm, which is quite accurate. It demonstrates the suitability of using the inverse rendering results for constructing the training data. On the other hand, our CoarseNet+FineNet achieves an average distance of 2.08 mm, which is comparable to that of the inverse rendering but with much faster processing speed (25 ms versus 8 s per frame). Some samples are shown in Fig. 10. In addition, the reconstruction accuracy by CoarseNet+FineNet outperforms the one by CoarseNet alone. Since the face region containing wrinkles is only a small part of the whole face region, the difference is not significant since the accuracy statistics is computed over a large face region. By comparing the reconstruction accuracy on a small region that contains wrinkles, the improvement is more obvious, as shown in Fig. 11.

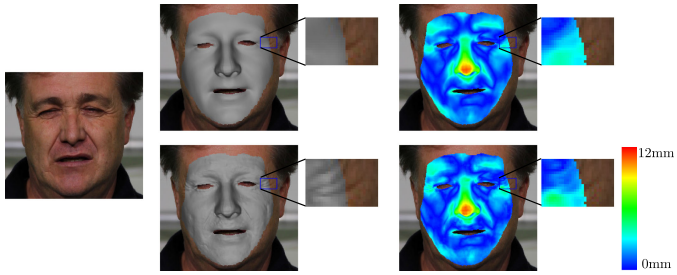


Fig. 11. Comparison of our CoarseNet and FineNet on a small region that is rich of wrinkles. On the left is the input frame, on the top are results of CoarseNet, on the bottom are results of FineNet. On the subregion, the mean error is 2.20 mm for CoarseNet and 2.03 mm for FineNet.

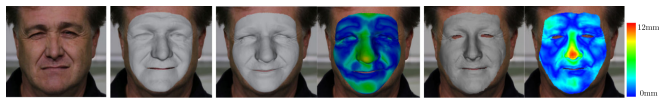


Fig. 12. The reconstruction accuracy comparison. The reconstruction quality of our dense face tracking method is comparable to the optimization based method [18] but with much faster processing speed. The groundtruth mesh is constructed using the binocular approach [55].

For the quantitative comparison with the state-of-the-art monocular video based face tracking method [18], we evaluate the geometric accuracy of the reconstruction of a video frame with rich face details (note that [18] did not provide the results for the entire video). Fig. 12 shows the results, where our method achieves a mean error of 1.96 mm compared to the groundtruth 3D face shape generated by the binocular facial performance capture proposed in [55]. We can see that the result of our learning based face tracking method is quite close to the groundtruth, and is comparable (1.96 mm versus 1.8 mm) to that of the complex optimization based approach [18] but with much faster processing speed.

9.2 Results of Dense 3D Face Reconstruction from A Single Image

Visual Results of our Single-Image Based Reconstruction. To evaluate the single-image based reconstruction performance, we show the reconstruction results of our method (Single-image CoarseNet+FineNet) on some images from AFLW [32] dataset, VGG-Face dataset [36] and some face images downloaded from internet. The three rows in Fig. 13 from top to bottom respectively show the projected 3D meshes reconstructed by our method under large poses, extreme expressions and face images with detailed wrinkles, which demonstrate that our method is robust to all of them.

Comparisons with Inverse Rendering. Similar to the video input scenario, directly using our developed inverse rendering approach can also reconstruct detailed geometries from a single image, but our learning-based method does provide some advantages. First, unlike the inverse rendering approach, our learning-based method does not need face alignment information. Therefore, the learning-based method is more robust to input face image with large pose, as shown in Fig. 14. Second, once the two CNNs are trained, our learning method is much faster to reconstruct a face geometry from a single input image. Third, as we render the same type of wrinkles under different lightings and directly learn the geometry in a supervised manner, our method is more

Fig. 13. For each pair, on the left is the input face image; on the right is the projected 3D mesh reconstructed by our single-image based solution. The first, second and third rows respectively demonstrate that our method is robust to large poses, extreme expressions and different types and scales of wrinkles.

Fig. 14. From left to right: input face image with detected landmarks, geometry reconstructed by inverse rendering, geometry reconstructed by our learning based method. It can be seen that our inverse rendering approach fails to recover the face shape as the landmarks are not accurate. On the other hand, our proposed learning-based approach recovers the face shape well.

Fig. 15. From left to right: input face image, geometry reconstructed by inverse rendering, geometry reconstructed by our learning based method. It can be seen that our method can better reconstruct unclear wrinkles under strong lighting.

robust to lighting, as illustrated in Fig. 15. The reason why the learning based method can do better in these scenarios lies in the large numbers of diverse training data we construct, which facilitate the learning of the two networks, while the inverse rendering approach only explores the information from each single image.

Comparisons with State-of-the-Art Single-Image Based Face Reconstruction. We compare our method with [3], [14], [24], [42], [46] on single-image based face reconstruction. We thank the authors of [42] for providing us the same 11 images listed in [42], as well as their results of another 8 images supplied by us. We show the reconstruction results of 4 images in Fig. 16 and the full comparisons on all the 19 images are

Fig. 16. Comparisons with the state-of-art methods. From the first row to the last row, it respectively shows the input images, and the results of [42], [3], [24], [46], [14] and ours. It can be seen that our results are more convincing in both the global geometry and the fine-scale details. Note that the method of [3] uses a 3DMM with identity variation only, and thus is not able to handle facial expressions well.

given in the accompanying material. It can be observed that our method produces more convincing reconstruction results in both the global geometry (see the mouth regions) and the fine-scale details (see the forehead regions). The reconstruction results of the methods [3], [14], [24], [46] are generated using the source codes provided by the authors.[2,3,4,5]

The reasons why our method produces better results than [42] are threefold: 1) For CoarseNet training, [42] only renders face region and uses random background, while our rendering is based on real images and the synthesized images are more photo-realistic. For FineNet training, we render images with fine-scale details, and train FineNet in a

2. https://github.com/waps101/3DMM_edges
3. https://github.com/AaronJackson/vrn
4. https://github.com/unibas-gravis/basel-face-pipeline
5. https://github.com/unibas-gravis/scalismo-faces

TABLE 2
Comparisons of Testing Errors Under Different Metrics

Metrics	[42]	Our method
$\mathcal{L}_{\text{pose}}$ in Eq. (13)	26.35	7.69
$\mathcal{L}_{\text{geo}}$ in Eq. (14)	5.53	4.23
MSE (pose parameters)	1.91	0.56
Mean vertex distance (geometry parameters)	5.18	4.55

supervised manner, while [42] trains FineNet in an unsupervised manner. 2) For easy back propagation, [42] adopts the first-order spherical harmonics to model lighting, while we use the second-order SH, which can reconstruct more accurate geometry details. 3) Our proposed loss function in CoarseNet better fits the goal and calculating the parameters in pixel level can achieve more stable and faster convergence. We did an experiment to compare our loss function $\mathcal{L}$ in Eq. (17) with the one used in [42]. Specifically, we used the two loss functions separately to train CoarseNet with 15,000 iterations and batch size 100. Table 2 shows the results of the test errors under different metrics on the test set (about 700 AFLW images). We can see that no matter which metric is used, either our defined metrics ($\mathcal{L}_{\text{pose}}$ and $\mathcal{L}_{\text{geo}}$), or the metrics employed in [42] (MSE for pose parameters and vertex distance for geometry parameters), our method always achieves lower testing errors than [42], which demonstrates the effectiveness of the defined loss function for training.

Quantitative Results of Single-Image Based Dense Face Reconstruction. For quantitative evaluation, we compare our method with the landmark-based method [61] and the learning-based method [60] on the Spring2004range subset of Face Recognition Grand Challenge dataset V2 [38]. The Spring2004range has 2,114 face images and their corresponding depth images. We use the face alignment method [28] to detect facial landmarks as the input of [61]. For comparison, we project the reconstructed 3D face on the depth image, and use both Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) metrics to measure the difference between the reconstructed depth and the ground truth depth on the valid pixels. We discard some images in which the projected face regions are very far away from the the real face regions for any of the three methods, which leads to a final 2,100 images being chosen for the comparisons. The results are shown in Table 3. It can be seen that our method outperforms the other two recent methods in both RMSE and MAE. The results of [61] and [60] are generated by directly running their released codes in public.

Note that we are not able to perform a quantitative comparison with the state-of-the-art method [42], since their code is not released. Their reported MAE value for the Spring2004range dataset is lower than what we obtain in Table 3. We believe it is due to the masks they used in their MAE computation, which are unfortunately not available to us. Although we cannot give a quantitative comparison, the visual comparison shown in Fig. 16 clearly demonstrates the superior face reconstruction performance of our method.

10 Conclusions

We have presented a coarse-to-fine CNN framework for real-time textured dense 3D face reconstruction and tracking from monocular RGB video as well as from a single RGB image. The training data to our convolutional networks are constructed by the optimization based inverse rendering approach. Particularly, we construct the training data by varying the pose and expression parameters, detail transfer as well as simulating the video-type adjacent frame pairs. With the well constructed large-scale training data, our framework recovers the detailed geometry, albedo, lighting, pose and projection parameters in real-time. We believe that our well constructed datasets including 2D face images, 3D coarse face models, 3D fine-scale face models, and multi-view face images of the same person could be applied to many other face analysis problems like face pose estimation, face recognition and face normalization.

Our work has limitations. Particularly, like many recent 3D face reconstruction works [18], [23], [48], we assume Lambertian surface reflectance and smoothly varying illumination in our inverse rendering procedure, which may lead to inaccurate fitting for face images with specular reflections or self-shadowing. It is worth to investigate more powerful formulation to handle general reflectance and illumination.

TABLE 3
Quantitative Comparison

Method	RMSE [mm]	MAE [mm]
[61]	5.946	4.420
[60]	5.367	3.923
Ours	4.915	3.846

Our method outperformes [61] and [60] in terms of RMSE and MAE.

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. The research is also partially supported by a grant (M4082186) for Joint WASP/NTU and MOE Tier-2 Grant (2016-T2-2-065).

References

[1] O. Aldrian and W. A. Smith, "Inverse rendering of faces with a 3D morphable model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 5, pp. 1080–1093, May 2013.
[2] B. Amberg, A. Blake, A. Fitzgibbon, S. Romdhani, and T. Vetter, "Reconstructing high quality face-surfaces using model based stereo," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2007, pp. 1–8.
[3] A. Bas, W. A. Smith, T. Bolkart, and S. Wuhrer, "Fitting a 3D morphable model to edges: A comparison between hard and soft correspondences," in *Proc. Asian Conf. Comput. Vis.*, 2016, pp. 377–391.
[4] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *Proc. 26th Annu. Conf. Comput. Graph. Interactive Techn.*, 1999, pp. 187–194.
[5] V. Blanz and T. Vetter, "Face recognition based on fitting a 3D morphable model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 9, pp. 1063–1074, Sep. 2003.
[6] S. Bouaziz, Y. Wang, and M. Pauly, "Online modeling for realtime facial animation," *ACM Trans. Graph.*, vol. 32, no. 4, 2013, Art. no. 40.
[7] C. Cao, D. Bradley, K. Zhou, and T. Beeler, "Real-time high-fidelity facial performance capture," *ACM Trans. Graph.*, vol. 34, no. 4, 2015, Art. no. 46.

[8] C. Cao, Q. Hou, and K. Zhou, "Displaced dynamic expression regression for real-time facial tracking and animation," *ACM Trans. Graph.*, vol. 33, no. 4, 2014, Art. no. 43.

[9] C. Cao, Y. Weng, S. Lin, and K. Zhou, "3D shape regression for real-time facial animation," *ACM Trans. Graph.*, vol. 32, no. 4, 2013, Art. no. 41.

[10] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou, "FaceWarehouse: A 3D facial expression database for visual computing," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 3, pp. 413–425, Mar. 2014.

[11] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Process.*, 2008, pp. 3869–3872.

[12] K. Dale, K. Sunkavalli, M. K. Johnson, D. Vlasic, W. Matusik, and H. Pfister, "Video face replacement," *ACM Trans. Graph.*, vol. 30, no. 6, 2011, Art. no. 130.

[13] B. Egger, A. Schneider, C. Blumer, S. Schönborn, and T. Vetter, "Occlusion-aware 3D morphable face models," in *Proc. Brit. Mach. Vis. Conf.*, 2016, pp. 64.1–64.11.

[14] B. Egger, S. Schönborn, A. Schneider, A. Kortylewski, A. Morel-Forster, C. Blumer, and T. Vetter, "Occlusion-aware 3D morphable models and an illumination prior for face image analysis," *Int. J. Comput. Vis.*, pp. 1–19, 2018.

[15] R. Garg, A. Roussos, and L. Agapito, "Dense variational reconstruction of non-rigid surfaces from monocular video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 1272–1279.

[16] P. Garrido, L. Valgaerts, O. Rehmsen, T. Thormahlen, P. Perez, and C. Theobalt, "Automatic face reenactment," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 4217–4224.

[17] P. Garrido, L. Valgaerts, C. Wu, and C. Theobalt, "Reconstructing detailed dynamic face geometry from monocular video," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 158:1–158:10, 2013.

[18] P. Garrido, M. Zollhöfer, D. Casas, L. Valgaerts, K. Varanasi, P. Pérez, and C. Theobalt, "Reconstruction of personalized 3D face rigs from monocular video," *ACM Trans. Graph.*, vol. 35, no. 3, 2016, Art. no. 28.

[19] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker, "Multi-pie," *Image Vis. Comput.*, vol. 28, no. 5, pp. 807–813, 2010.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[21] 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, 2011, Art. no. 74.

[22] P. Huber, G. Hu, R. Tena, P. Mortazavian, P. Koppen, W. J. Christmas, M. Ratsch, and J. Kittler, "A multiresolution 3D morphable face model and fitting framework," in *Proc. 11th Int. Joint Conf. Comput. Vis. Imaging Comput. Graph. Theory Appl.*, 2016.

[23] A. E. Ichim, S. Bouaziz, and M. Pauly, "Dynamic 3D avatar creation from hand-held video input," *ACM Trans. Graph.*, vol. 34, no. 4, 2015, Art. no. 45.

[24] A. S. Jackson, A. Bulat, V. Argyriou, and G. Tzimiropoulos, "Large pose 3D face reconstruction from a single image via direct volumetric CNN regression," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1031–1039.

[25] 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 *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[26] L. Jiang, J. Zhang, B. Deng, H. Li, and L. Liu, "3D face reconstruction with geometry details from a single image," *CoRR*, vol. abs/1702.05619, 2017.

[27] A. Jourabloo and X. Liu, "Large-pose face alignment via CNN-based dense 3D model fitting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4188–4196.

[28] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1867–1874.

[29] 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.*, vol. 33, no. 2, pp. 394–405, Feb. 2011.

[30] I. Kemelmacher-Shlizerman, A. Sankar, E. Shechtman, and S. Seitz, "Being john malkovich," in *Proc. Comput. Vis.*, 2010, pp. 341–353.

[31] H. Kim, M. Zollhöfer, A. Tewari, J. Thies, C. Richardt, and C. Theobalt, "InverseFaceNet: deep monocular inverse face rendering," *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.

[32] M. Köstinger, P. Wohlhart, P. M. Roth, and H. Bischof, "Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2011, pp. 2144–2151.

[33] H. Li, J. Yu, Y. Ye, and C. Bregler, "Realtime facial animation with on-the-fly correctives," *ACM Trans. Graph.*, vol. 32, no. 4, Jul. 2013, Art. no. 42.

[34] F. Liu, D. Zeng, Q. Zhao, and X. Liu, "Joint face alignment and 3D face reconstruction," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 545–560.

[35] C. Müller, "Spherical harmonics," in *Lecture Notes in Mathematics*, Berlin, Germany: Springer, 1966.

[36] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proc. Brit. Mach. Vis. Conf.*, 2015, pp. 41.1–41.12.

[37] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, "A 3D face model for pose and illumination invariant face recognition," in *Proc. IEEE Int. Conf. Advanced Video Signal Based Surveillance*, 2009, pp. 296–301.

[38] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek, "Overview of the face recognition grand challenge," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 947–954.

[39] E. Prados and O. Faugeras, "Shape from shading," in *Handbook of Mathematical Models in Computer Vis.*. Berlin, Germany: Springer, 2006, pp. 375–388.

[40] R. Ramamoorthi and P. Hanrahan, "An efficient representation for irradiance environment maps," in *Proc. 28th Annu. Conf. Comput. Graph. Interactive Techn.*, 2001, pp. 497–500.

[41] E. Richardson, M. Sela, and R. Kimmel, "3D face reconstruction by learning from synthetic data," in *Proc. Int. Conf. 3D Vis.*, 2016, pp. 460–469.

[42] E. Richardson, M. Sela, R. Or-El , and R. Kimmel, "Learning detailed face reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 5553–5562.

[43] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 2015, pp. 234–241.

[44] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: The first facial landmark localization challenge," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2013, pp. 397–403.

[45] S. Saito, T. Li, and H. Li, "Real-time facial segmentation and performance capture from RGB input," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 244–261.

[46] S. Schönborn, B. Egger, A. Morel-Forster, and T. Vetter, "Markov chain Monte Carlo for automated face image analysis," *Int. J. Comput. Vis.*, vol. 123, no. 2, pp. 160–183, 2017.

[47] J. Shen, S. Zafeiriou, G. G. Chrysos, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, "The first facial landmark tracking in-the-wild challenge: Benchmark and results," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop*, 2015, pp. 1003–1011.

[48] F. Shi, H.-T. Wu, X. Tong, and J. Chai, "Automatic acquisition of high-fidelity facial performances using monocular videos," *ACM Trans. Graph.*, vol. 33, no. 6, 2014, Art. no. 222.

[49] I. Shimshoni, Y. Moses, and M. Lindenbaum, "Shape reconstruction of 3D bilaterally symmetric surfaces," *Int. J. Comput. Vis.*, vol. 39, no. 2, pp. 97–110, 2000.

[50] G. W. Stewart, "On the early history of the singular value decomposition," *SIAM Rev.*, vol. 35, no. 4, pp. 551–566, 1993.

[51] A. Tewari, M. Zollöfer, H. Kim, P. Garrido, F. Bernard, P. Perez, and T. Christian, "MoFA: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3735–3744.

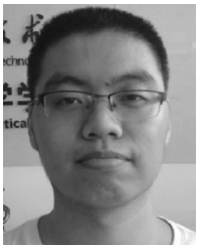
[52] J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt, "Real-time expression transfer for facial reenactment," *ACM Trans. Graph.*, vol. 34, no. 6, 2015, Art. no. 183.

[53] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of RGB videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2387–2395.

[54] A. T. Tran, T. Hassner, I. Masi, and G. Medioni, "Regressing robust and discriminative 3D morphable models with a very deep neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1493–1502.

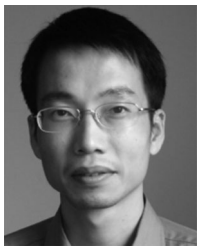
[55] L. Valgaerts, C. Wu, A. Bruhn, H. Seidel, and C. Theobalt, "Lightweight binocular facial performance capture under uncontrolled lighting," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 187:1–187:11, 2012.

[56] T. Weise, S. Bouaziz, H. Li, and M. Pauly, "Realtime performance-based facial animation," *ACM Trans. Graph.*, vol. 30, 2011, Art. no. 77.
[57] L. Williams, "Performance-driven facial animation," *ACM SIGGRAPH Comput. Graph.*, vol. 24, pp. 235–242, 1990.
[58] W. Y. Zhao and R. Chellappa, "Illumination-insensitive face recognition using symmetric shape-from-shading," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2000, pp. 286–293.
[59] W. Y. Zhao and R. Chellappa, "Symmetric shape-from-shading using self-ratio image," *Int. J. Comput. Vis.*, vol. 45, no. 1, pp. 55–75, 2001.
[60] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li, "Face alignment across large poses: A 3D solution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 146–155.
[61] X. Zhu, Z. Lei, J. Yan, D. Yi, and S. Z. Li, "High-fidelity pose and expression normalization for face recognition in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 787–796.

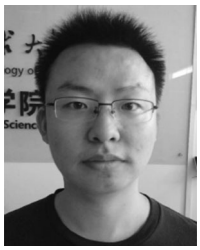


Yudong Guo received the bachelor's degree from the University of Science and Technology of China, in 2015. He is working toward the master's degree in the School of Mathematical Sciences, University of Science and Technology of China. His research interests include computer vision and computer graphics.

Juyong Zhang received the BS degree from the University of Science and Technology of China, in 2006, and the PhD degree from Nanyang Technological University, Singapore. He is an associate professor with the School of Mathematical Sciences, University of Science and Technology of China. His research interests include computer graphics, computer vision, and numerical optimization. He is an associate editor of the *Visual Computer*.



Jianfei Cai (S'98-M'02-SM'07) received the PhD degree from the University of Missouri-Columbia. He is currently an associate professor and has served as the head of Visual & Interactive Computing Division and the head of Computer Communication Division with the School of Computer Engineering, Nanyang Technological University, Singapore. His major research interests include multimedia, computer vision and visual computing. He has published more than 200 technical papers in international journals and conferences. He is currently an AE for the *IEEE Transactions on Multimedia*, and has served as an AE for the *IEEE Transactions on Image Processing* and the *IEEE Transactions on Circuits and Systems for Video Technology*. He is a senior member of the IEEE.



Boyi Jiang received the bachelor's degree in mathematical sciences from the University of Science and Technology of China, in 2015. He is working toward the master's degree in the same University. His research interests are computer vision and digital geometry processing.

Jianmin Zheng received the BS and PhD degrees from Zhejiang University, China. He is an associate professor with the School of Computer Engineering, Nanyang Technological University, Singapore. His recent research focuses on T-spline technologies, digital geometric processing, reality computing, 3D vision, interactive digital media and applications. He has published more than 150 technical papers in international conferences and journals. He was the conference co-chair of Geometric Modeling and Processing 2014 and has served on the program committee of several international conferences. He is an associate editor of the *Visual Computer*.