Real-time Hierarchical Facial Performance Capture

Luming Ma
University of Houston
lma15@uh.edu

Zhigang Deng
University of Houston
zdeng4@uh.edu

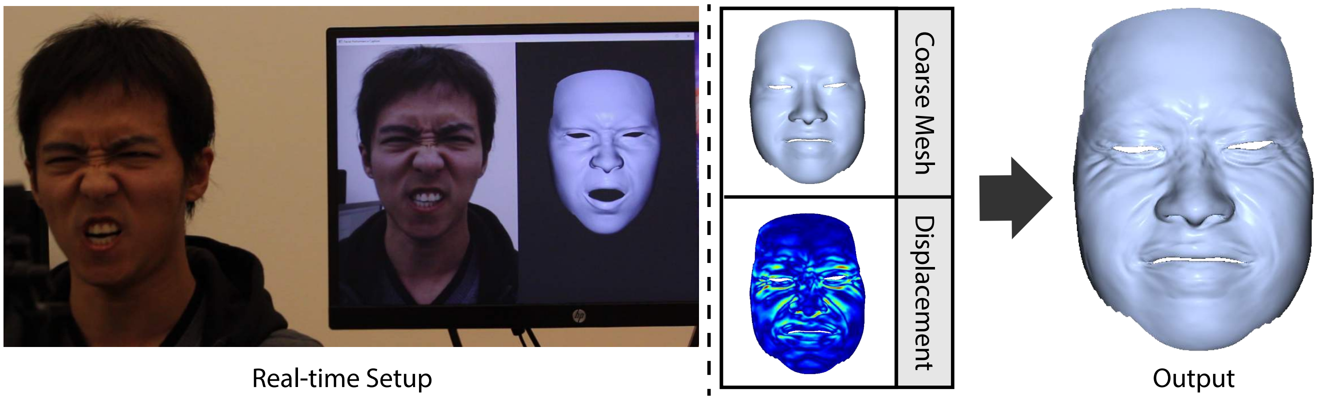


Figure 1: Our system captures fine-scale facial performance from a monocular RGB camera in real-time (left). We augment the large-scale facial performance by per-vertex displacements via shape-from-shading to capture wrinkle-level details (right).

ABSTRACT

This paper presents a novel method to reconstruct high resolution facial geometry and appearance in real-time by capturing an individual-specific face model with fine-scale details, based on monocular RGB video input. Specifically, after reconstructing the coarse facial model from the input video, we subsequently refine it using shape-from-shading techniques, where illumination, albedo texture, and displacements are recovered by minimizing the difference between the synthesized face and the input RGB video. In order to recover wrinkle level details, we build a hierarchical face pyramid through adaptive subdivisions and progressive refinements of the mesh from a coarse level to a fine level. We both quantitatively and qualitatively evaluate our method through many experiments on various inputs. We demonstrate that our approach can produce results close to off-line methods and better than previous real-time methods.

CCS CONCEPTS

• **Computing methodologies → Shape modeling**; *Image-based rendering*;

KEYWORDS

Real-time 3D face reconstruction, multi-resolution modeling, per-vertex displacements, shape-from-shading

ACM Reference Format:
Luming Ma and Zhigang Deng. 2019. Real-time Hierarchical Facial Performance Capture. In *Symposium on Interactive 3D Graphics and Games (I3D '19), May 21–23, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3306131.3317016

1 INTRODUCTION

Capturing human facial performance has been a long standing problem in computer graphics [Zollhöfer et al. 2018]. Since the human eyes are particularly sensitive to facial performance and appearance, creation of high resolution 3D face models and realistic facial animations for films, VR systems, and games often involves controlled lighting setups, multiple cameras, active markers, and substantial post-editing from experienced artists [Williams 2006]. In recent years, social media and mobile applications bring increasing demands for light-weight acquisition of facial performance on consumer devices. To this end, passive methods leveraging stereo cameras [Valgaerts et al. 2012] or depth information [Li et al. 2013; Weise et al. 2011] have been proposed to capture detailed facial performance using binocular or RGB-D cameras. These methods achieve impressive results but are limited to the requirement of binocular footages or depth data which are often unavailable for legacy video footages.

Recently, researchers sought to only rely on monocular RGB video for facial reconstruction [Garrido et al. 2013, 2016; Shi et al. 2014]. These methods utilize shading information for shape refinements and achieve quality on par with the methods with stereo and depth cameras. In the meantime, these methods require substantial computational time as well as information from forward

I3D '19, May 21–23, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6310-5/19/05…$15.00
https://doi.org/10.1145/3306131.3317016

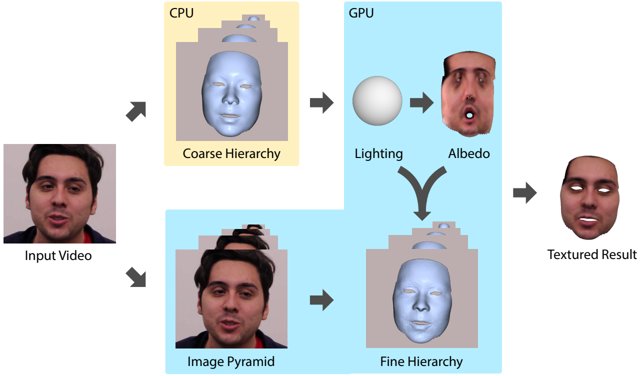


Figure 2: System overview. Our system takes RGB video as input and builds coarse mesh layers and image layers for hierarchical reconstruction (Section 3). It then computes lighting (Section 4.1), albedo (Section 4.2), and vertex displacements (Section 4.3) using shape-from-shading techniques. The output is a high resolution mesh with wrinkle-level details.

and backward frames of the video; therefore, they are unsuitable for real-time acquisition applications. On the other hand, real-time facial tracking and animation systems have also been developed with a single RGB camera [Cao et al. 2014a, 2013]. Their data-driven methods, however, rely on strong face priors and could not capture individual-specific or transient details, such as wrinkles on the forehead and around the eyes. Their recent development [Cao et al. 2015] incorporates medium-scale frequencies into the coarse face model by training regressors from high-resolution facial capture data. Even though plausible face wrinkles are produced, in a nutshell, this learning-based (or data-driven) model is yet an approximation of the true facial performance and is also limited by the data used for training. Despite considerable advances in facial performance tracking, a method that accurately captures facial details, with low cost acquisition (e.g., a monocular camera) and in a real-time rate, is still demanding.

In this paper we propose a geometry-based method to reconstruct high resolution facial geometry and appearance in real-time (Figure 1). It takes in RGB video from a single RGB camera and captures an individual-specific face model of the subject with wrinkle-level details. The system runs fully automatic and does not require off-line preprocessing for a novel user. We show that our results are close to existing off-line methods. We believe this opens up more possibilities in consumer-level interactive applications, such as facial performance avatar retargeting, on-line preview, immersive VR games, and photo-realistic facial makeups.

Our method captures fine-scale facial performance by estimating per-vertex displacements. We first reconstruct a low resolution facial geometry from the input video. This model presents large-scale facial expressions but lacks wrinkle level details. We subsequently refine it using shape-from-shading techniques. The coefficients of incident illumination, albedo texture, and displacements are recovered by minimizing the difference between the synthesized face and the input RGB images. To endow the face model the capability to capture fine-scale details, we adaptively subdivide the mesh into a hierarchy of multiple resolutions. We also build a corresponding pyramid for the input images. The vertex displacements estimated on a coarse mesh capture low frequency deformations and are prolonged to finer levels, where higher frequency details will be captured from images of higher resolutions. Figure 2 shows the schematic pipeline of our system.

The main contributions of our work can be summarized as:

- a complete, fully automatic system to capture fine-scale facial performance from monocular RGB video;
- a hierarchical reconstruction method that is efficient and robust to reconstruct high resolution face models; and
- a novel vertex displacement formula to solve the shape-from-shading problem.

2 RELATED WORK

Off-line Face Reconstruction. Many previous works have been focused on building 3D face models in controlled environments [Klehm et al. 2015]. The works of [Ma et al. 2008; Zhang et al. 2004] employ structured light and photometric stereo for face scanning. Some methods attach markers on the face [Bickel et al. 2007; Huang et al. 2011] to acquire dynamic deformations of 3D facial performance. Despite high quality face models reconstructed, the above methods typically involve complex intrusive setups to the subjects. Passive solutions were also developed using stereo images [Beeler et al. 2010, 2011; Bradley et al. 2010; Gotardo et al. 2018] or lightweight binocular cameras [Valgaerts et al. 2012]. Some other works rely on data priors to constrain the reconstructed face, such as morphable models [Blanz and Vetter 1999] and multi-linear models [Cao et al. 2014b; Vlasic et al. 2005]. High frequency details such as wrinkles are typically missing from these methods, which requires further refinements [Bermano et al. 2014].

Reconstruction from monocular video [Fyffe et al. 2014; Garrido et al. 2013; Shi et al. 2014; Suwajanakorn et al. 2014] in uncontrolled environments draws more attention in recent years due to its low cost setup and applicability to various legacy video footage. Follow-up works of [Garrido et al. 2016; Suwajanakorn et al. 2015] build controllable face rigs and appearance from video for animation. Similarly, Ichim et al. [2015] create fully rigged 3D personalized avatars from hand-held video input. All the above methods, however, require intensive off-line processing and are not applicable to real-time scenarios.

Real-time Face Capture. Real-time facial tracking systems have been developed using structured light [Weise et al. 2009], where an individual-specific face model is fitted offline first and then tracked online for expression transferring. Another category combines depth information from a single RGB-D camera [Chen et al. 2013; Hsieh et al. 2015; Li et al. 2013; Thies et al. 2015, 2018a,b; Weise et al. 2011; Zollhöfer et al. 2014] to track facial expression deformations in real-time. Regression based face tracking has been proposed by Cao et al. [2014a, 2013] to capture coarse 3D facial geometry from a single monocular camera in real-time. Their follow-up work [Cao et al. 2015] learns displacement patches from captured texture to predict medium-scale details. Recently, Wang et al. [2016] track facial expression and eye gaze simultaneously, and Thies et al. [2016] fit parametric face models by combining the photometric

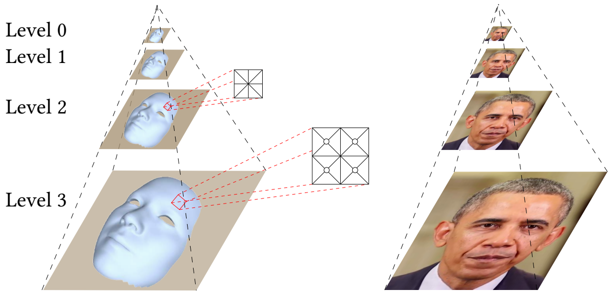


Figure 3: An example of 4-level hierarchy where lower levels have higher resolution meshes and images. The closeups show the topology for the same mesh blocks at two consecutive levels, with the lower level containing 4 more vertices denoted as circles. ©public domain.

consistencies from RGB input sequence. From the perspective of computer vision community, deep learning methods, such as CNN [Guo et al. 2018; Sela et al. 2017; Tewari et al. 2018, 2017], have been extensively used to reconstruct facial performance from images/video. Even though noticeable progresses have been made in the area of real-time face capture, fine-scale details reconstruction is still a weakness compared to off-line methods.

Shape-from-Shading. Acquiring 3D shape from a single image given the known lighting condition and surface reflectance is a well established technique, a.k.a shape-from-shading (SfS) [Horn 1975], in the area of photometric stereo. Vlasic et al. [2009] capture dynamic normal maps from multiple views using SfS under designed lighting. A popular use case of SfS is coarse shape refinements under uniform lighting [Beeler et al. 2012] or even uncontrolled lighting [Wu et al. 2013, 2011] by approximating lighting with spherical harmonics [Basri and Jacobs 2003]. This technique has also been widely used in facial reconstruction for fine-scale geometry refinements since human faces are generally assumed to be Lambertian surfaces with statistical shape priors. The works of [Garrido et al. 2013, 2016; Ichim et al. 2015; Kemelmacher-Shlizerman and Basri 2011; Shi et al. 2014; Suwajanakorn et al. 2014] reconstruct fine-scale face shapes and albedo from RGB video in an off-line manner. A real-time method related to ours is [Wu et al. 2014], where lighting and refined depth map are estimated from shading cues using a RGB-D camera. Their method, however, cannot reconstruct parametric face models or albedo texture, while our method builds a textured face blendshape model that is ready for animation and expression reenactment using only a monocular camera.

3 HIERARCHICAL RECONSTRUCTION

Our method focuses on augmenting a coarse face model captured from input RGB video with wrinkle and fold details. The coarse model could be generated by any real-time methods such as [Cao et al. 2014a] and [Wang et al. 2016]. In our system, we use [Wang et al. 2016] which fits a bilinear face model to the input video by estimating the camera parameters, head pose, identity vector, and expression vector. The captured face model contains 5.6K vertices and 33K triangle faces presenting large-scale facial deformations.

Mesh and Image Hierarchy. The resolution of the mesh is too low to faithfully reproduce the subtle details of the input image through vertex displacements. Therefore, we propose a hierarchical approach to progressively capture finer details using higher resolution meshes. Specifically, we use the coarse mesh as the control mesh and iteratively subdivide it to build a mesh hierarchy $\{M_0, M_1, \ldots, M_k\}$ with M_0 denoting the coarsest level and M_k denoting the finest level. A corresponding image pyramid $\{I_0, I_1, \ldots, I_k\}$ is also constructed from the GPU generated mipmap of the input frame. Figure 3 shows an example of the constructed hierarchy, where the top level has the lowest resolution mesh and image while the bottom level has the highest resolution. The closeups show that 4 new vertices (circle) are inserted into a mesh block (red square) after subdivision to account for the additional pixels in the corresponding higher resolution image.

4-8 subdivision. We employ the 4-8 subdivision scheme [Velho and Zorin 2001] to adaptively subdivide the mesh. The reason we choose this scheme is two-folds: (i) our coarse mesh M_0 is a triangulated quadrilateral mesh which exactly meets the requirement of 4-8 subdivision. (ii) 4-8 subdivision results in conforming meshes (without edge cracks) and simple adaptive subdivisions. As shown in Figure 4, our initial control mesh consists of thousands of quad blocks. The two triangles $\{f_1, f_2\}$ are called the *mate* triangles of a block $\{v_1, v_2, v_3, v_4\}$. Notice that a regular triangle f_1 has two vertices v_1, v_3 with valence = 8 and one vertex v_2 with valence = 4. We denote the edge v_1v_3 which is opposite to the vertex with valence = 4 as an interior edge. The basic subdivision operation of a mesh block is bisection: a new vertex is inserted to bisect the interior edge and two new edges are created to connect the new vertex to the two vertices with valence = 4. The positions of the new vertex and the old vertices are computed using the face mask and vertex mask, respectively, as shown in the middle of Figure 4.

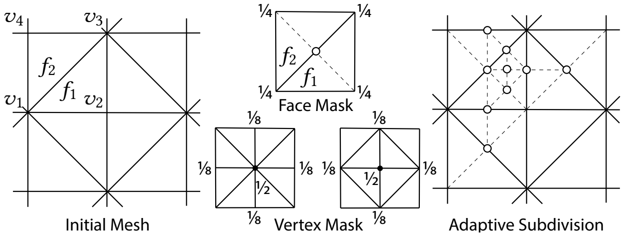


Figure 4: 4-8 subdivision. Left: a coarse initial mesh; Middle: the face/vertex mask for mesh block bisection; Right: the multi-level adaptive subdivision. The added vertices and edges are denoted as circles and dashed lines, respectively.

We build each finer mesh M^{i+1} from the coarser mesh M^i by looping through all the faces $f \in F$ with the *adaptFace* function (Algorithm 1). To prevent over-refinements, i.e., multiple vertices are projected to a single pixel, we skip the triangle faces whose interior edge is smaller than a threshold ε when projected to the image space (line 4) using the estimated head pose and camera parameters. The subdivision terminates when no triangle face needs to be adapted. Prior to the bisection of a face f, the *mate* face of f (denoted as $f.mate$) needs to be at the same subdivision level as f (line 5) to prevent cracks, since $f.mate$ might be skipped by previous subdivision steps. Then, we update the positions of the two end vertices of the interior edge of f using the *adaptVertex* function. The actual bisection operation of *adaptFace* is performed by bisecting the interior edge (line 10) and the two triangles f (line 12) and $f.mate$ (line 13). Note that we only construct the mesh hierarchy at the start of the video and then keep it fixed. At run-time, the mesh is progressively refined from coarse levels and prolonged to fine levels following the same topology and face/vertex mask. We will discuss our displacements-based hierarchical mesh refinements in more details in Section 4.3.

Algorithm 1: Adaptive 4-8 Subdivision

```
1  Function adaptFace($V, F, f$):
2      if $f$ has not been processed then
3          $e \leftarrow$ getInteriorEdge($f$);
4          if project(e).length $> \varepsilon$ then
5              while f.mate.level < f.level do
6                  adaptFace($V, F, f.mate$);
7              end
8              adaptVertex($V, F, e.start, f.level$);
9              adaptVertex($V, F, e.end, f.level$);
10             $v \leftarrow$ bisectEdge($e$);
11             put $v$ in $V$;
12             $f_1 \leftarrow$ bisectTriangle($f$);
13             $f_2 \leftarrow$ bisectTriangle($f.mate$);
14             put $f_1, f_2$ in $F$;
15         end
16     end
17 end
18 Function adaptVertex($V, F, v, l$):
19     if $v$ has not been processed then
20         for $f \in F$ containing $v$ do
21             while f.level < $l$ do
22                 adaptFace($V, F, f$);
23             end
24         end
25         updateVertexPosition($v$);
26     end
27 end
```

4 SHADING BASED REFINEMENTS

Similar to previous off-line methods [Shi et al. 2014], we assume human faces are Lambertian surfaces, and employ shape-from-shading to capture dynamic fine details such as wrinkles and folds. We encode the fine surface bumps as the displacements along surface normals and recover them jointly with the unknown illumination and albedo from the input RGB images. We also parameterize the incident lighting with spherical harmonics [Basri and Jacobs 2003], and assume the lighting contributes equally to each RGB channel. Therefore, the reflected irradiance R at vertex i is represented as:

$$R_i = l^T \cdot SH(n_i)\rho_i, \tag{1}$$

where ρ_i is the face albedo at vertex i, l is the vector of spherical harmonics coefficients of incident lighting, and SH is the spherical harmonics basis functions taking a unit length surface normal n_i as the input. We take the 2nd order harmonic approximation that captures more than 99% of the energy in a face image [Kemelmacher-Shlizerman and Basri 2011]. Then $SH(n) \in \mathbb{R}^9$ evaluates to:

$$SH(n) = (1, n_x, n_y, n_z, n_x n_y, n_x n_z, n_y n_z, n_x^2 - n_y^2, 3n_z^2 - 1)^T,$$

where n_x, n_y, and n_z are the components of the normal n.

We solve the inverse rendering equation (Eq. 1) in an analysis-by-synthesis strategy. The lighting, albedo, and displacements are iteratively optimized by minimizing the difference between the current observed image I and the rendered image R:

$$E_{data} = \sum_{i=1}^{K} \|I_i - R_i\|^2, \tag{2}$$

where I_i is the sampled image color by projecting vertex i onto the image plane according to head pose and camera parameters, and K is the number of vertices. At a frame t, we first roughly approximate the refined (target) face mesh as the captured coarse mesh augmented with the displacements from the previous frame:

$$M_d^t = M^t + d^{t-1} N^t,$$

where M^t is the coarse mesh without refinements, N^t is its derived per-vertex normals, and M_d^t is the augmented mesh with the previous frame’s displacements d^{t-1}. We will drop the superscript t in the remainder of this paper. Then, the lighting can be estimated (Sec. 4.1) using the recomputed vertex normals from M_d and the albedo from the previous frame. The albedo is subsequently updated (Sec. 4.2) with the computed lighting and normals. Note that the albedo is computed only at the start of the video and remains fixed thereafter. Finally, the displacements are re-estimated (Sec. 4.3) for current frame by formulating normals as a function of displacements. We describe the details of each step below.

4.1 Lighting Estimation

We assume the illumination varies across frames but penalize sudden changes between consecutive frames with a smoothing term. The total energy function for illumination becomes:

$$E(l) = E_{data} + w_1 \left\| l^t - l^{t-1} \right\|^2, \tag{3}$$

Minimizing this energy in terms of l is equivalent to solving the linear system $Al = b$, where A is a $(3K + 9)$ by 9 matrix. The top $3K$ rows evaluate to the product of $SH(n)$ and each channel of ρ, and the bottom 9 rows are an identity matrix. We only consider those visible vertices whose normals face to the camera ($n_z > 0$). This leads to a highly over-constrained linear system and substantial set-up and computational time if conducted on the CPU. To achieve real-time performance, we resort to a pure GPU solution. We construct the matrix A and the vector b on the GPU with each thread assembling one row. Then, we compute $A^T A$ and $A^T b$ using cuBLAS and solve the *normal equation* $A^T Ab = A^T b$ using Cholesky decomposition provided by cuSOLVER. Since illumination is a global environment attribute, its accuracy will not be significantly improved on finer levels. Hence, we directly use the coarsest mesh M_0 for fast computation.

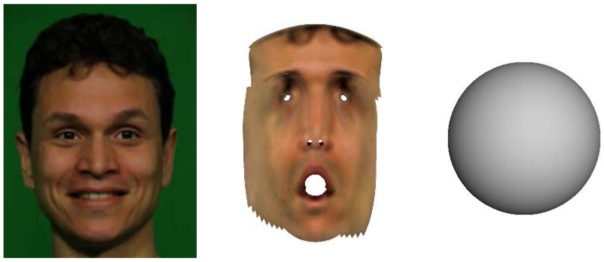


Figure 5: From an input RGB image (left), we recover albedo texture (middle) and incident lighting (right).

4.2 Albedo Recovery

Given the estimated lighting and approximated normals, the albedo at each vertex can be naively computed as $\rho_i = \frac{I_i}{l^T \cdot SH(n_i)}$. However, this leads to baking the residuals of E_{data} into the albedo, such that fine details, such as wrinkles, are interpreted as albedo change. Therefore, we incorporate a Laplacian regularization term to adapt the albedo to be as smooth as the prior average albedo. The energy function becomes:

$$E(\rho) = E_{data} + w_2 \left\| L\rho - L\bar{\rho} \right\|^2 , \tag{4}$$

where $\bar{\rho}$ is the average albedo provided by the FaceWarehosue [Cao et al. 2014b], and L represents the graph Laplacian matrix with respect to the mesh.

Minimizing Eq. 4 is equivalent to solving a sparse linear least square problem $A\rho = b$. For a mesh with E edges, this problem has $3K$ unknowns, $(3K + 3E)$ residuals, and $(3K + 6E)$ non-zero values. Similar to the lighting estimation, we also convert it to the *normal equation* using cuSPARSER and solve it using cuSOLVER. Note that we only update the albedo within the first several frames and keep it fixed afterwards. Because we would like to elude baking geometric detail into the albedo, it is desired for all the mesh levels in our hierarchy to have as close as possible albedo. Therefore, we compute the albedo on the coarsest mesh M_0 and prolong to finer levels where the albedo for additional vertices are interpolated with the vertex mask in Figure 4. Figure 5 shows an example of the recovered albedo and lighting.

4.3 Displacements Refinement

Now we introduce how to augment the coarse mesh with geometric details via shape-from-shading. Equipped with the initialized mesh hierarchy described in Section 3, we first build the corresponding image pyramid on the GPU for a particular frame t. Starting from the coarsest level, we enhance the mesh with vertex displacements to improve the consistency between surface reflectance R and the corresponding image I. The computed displacements on coarser level are then prolonged to the next finer level as an initialization for further improvement as well as a constraint to regularize the displacements on the finer level to be close to those on the coarser level. The advantages of this progressive approach over direct refinements on the finest level include improved robustness and less noise. Therefore, to obtain a smooth face surface, a smoothness term that measures local displacement smoothness is required in the energy function. A proper weight of this term, however, is very difficult to find: a large weight may result in an over-smoothed surface that does not contain fine details; while a small weight may bring noise to the surface. In our approach, we use coarser levels to capture lower frequency bands of geometric features that serve as a reasonable guidance for refinements at finer levels. On finer levels, higher frequency details are added and noise is removed using the higher resolution input image. The difference between with and without hierarchical refinements can be seen in Figure 6: using a small smoothness weight, the direct reconstruction method incurs noise around nose; while our hierarchical method gradually removes noise from coarse levels and adds details on finer levels.

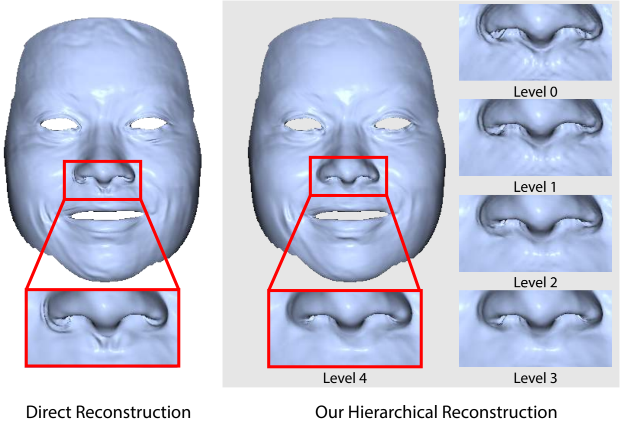


Figure 6: Our hierarchical method (right) progressively refines the coarse mesh, producing a less noisy, high resolution geometry, compared to the direct reconstruction method (left) that only refines on the highest resolution level.

To solve the displacements d, we first formulate E_{data} as a function of d. For a vertex v_i, its normal n_i is calculated as a weighted sum of all the neighboring face normals. If the weight ω is set to be twice of the area of the neighbor triangle, the vertex normal calculation can be converted to:

$$n_i = \sum \omega \frac{(v_j - v_i) \times (v_k - vi)}{|(v_j - v_i) \times (v_k - v_i)|} = \sum (v_j - v_i) \times (v_k - v_i),$$

where v_j and v_k are the other two vertices of a triangle containing v_i. In this way, we can obtain the vertex normals for the coarse mesh. Recall that the refined vertex position can be presented as a displacement along the coarse normal, thus the vertex normal on the desired refined mesh can be expressed as a function of displacements d:

$$n_i^* = \sum ((v_j + d_j n_j) - (v_i + d_i n_i)) \times ((v_k + d_k n_k) - (v_i + d_i n_i)), \tag{5}$$

where d_i, d_j, and d_k denote the unknown displacements for vertex v_i, v_j, and v_k, respectively. By substituting Eq. 5 into E_{data} (Eq. 2), we obtain an energy function where d is the only unknown variable. Nonetheless, the problem is still under-constrained. We instead impose additional constraints and optimize the following non-linear

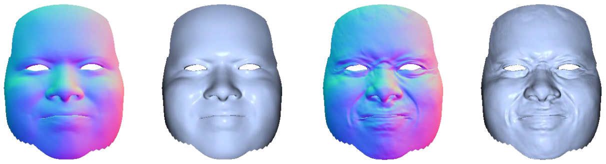


Figure 7: The normal map and coarse mesh before (left) and after (right) our shape-from-shading refinements.

energy function:

$$\begin{aligned} E(d) =& E_{data} + w_3 E_{smooth} + w_4 E_{hier} + w_5 E_{coarse}, \\ & \text{s.t.} \quad \partial M(d) = 0, \end{aligned} \tag{6}$$

where ∂M denotes the boundary of the face surface, serving as the Dirichlet boundary condition.

Smoothness Constraint. For a C^2 surface, its local displacements should change smoothly. Similar to albedo, we employ graph Laplacian to improve the displacement smoothness:

$$E_{smooth} = \|Ld\|^2 = \sum_{i=1}^{K} \left\| d_i - \sum_{j=1}^{r} \frac{1}{r} d_j \right\|^2 ,$$

where r is the number of 1-ring neighbors for vertex v_i, and d_j is the j-th neighbor's displacements.

Hierarchical Constraint. For the mesh at a level higher than 0, we initialize its displacements as the prolongation of the displacements at the coarser level. We also use the prolonged displacements pd as a regularizer for the current level:

$$E_{hier} = \sum_{i=1}^{K} \|pd_i - d_i\|^2 .$$

Coarse Constraint. We assume that the coarse mesh already provides a good approximation of the ground truth, thus the displacements are expected to be small:

$$E_{coarse} = \sum_{i=1}^{K} \|d_i\|^2 .$$

Figure 7 presents the enhanced surface details for the normal map and the geometry after refinements. The non-linear energy Eq. 6 consists of K unknowns and $6K$ residuals. We employ a data-parallel GPU Gauss-Newton solver for real-time optimization.

4.4 Energy Minimization

Gauss-Newton Solver. The non-linear energy $E(d)$ can be rewritten as the sum of squared residuals:

$$E(d) = \sum_{k=1}^{6K} \|\mathrm{r}_k(d)\|^2 = \sum_{k=1}^{6K} \|\mathrm{y}_k - \mathrm{f}_k(d)\|^2 . \tag{7}$$

Initialized by the displacements from the previous frame, we aim to find the local optimal displacements $d^* \in \mathbb{R}^K$ where the gradient is zero. To this end, we update the parameter vector d through several Gauss-Newton steps. At a step n, d^n is updated as:

$$d^{n+1} = d^n + \Delta d \quad \text{and} \quad J^T J \Delta d = J^T(\mathrm{y} - \mathrm{f}(d)), \tag{8}$$

where J is the Jacobian matrix of f with its $(i, j)^{\text{th}}$ component $= \frac{\partial \mathrm{f}_i(d)}{\partial d_j}$. We only evaluate the data term of J at the initialization of each step and store it into the global memory while evaluate the other terms on-the-fly for less memory access. The optimal step length Δd is computed by the PCG solver, described below.

Preconditioned Conjugate Gradient (PCG) Solver. Similar to the work of [Zollhöfer et al. 2014], our PCG solver requires 2 kernel calls at initialization and 3 kernel calls per iteration loop for synchronizations across thread blocks. We use the Jacobi conditioner to ensure a fast convergence. The inverses of the diagonal entries of $J^T J$ are computed at initialization. However, to take advantage of the sparsity of J, we never explicitly evaluate $J^T J$. Instead, whenever we need the multiplication of matrix $J^T J$ with a vector p, we convert it to two sequential matrix-vector multiplications Jp and $J^T(Jp)$.

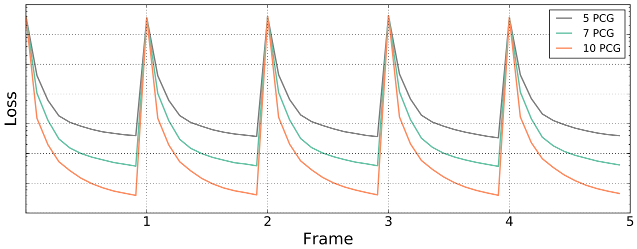


Figure 8: The convergences of our Gauss-Newton solver for 10 Gauss-Newton steps within 5 frames.

5 RESULTS

In this section, we first describe the implementation and runtime of our method. Then we both quantitatively and qualitatively evaluate our method by comparing our result with input video, ground-truth meshes, and results by existing offline and real-time methods.

5.1 Implementation

We implemented the coarse face reconstruction on the CPU using C++ and the hierarchical refinements on the GPU using CUDA. The two components run in fully parallel. We run 10 outer Gauss-Newton steps and 7 inner PCG iterations for displacements refinement. We compare the convergences of our solver with various configurations, as shown in Figure 8. We found that the 10/7 configuration achieved the optimal balance between the convergence rate and the computational cost.

Table 1 presents the runtime statistics of our system that runs on a desktop computer with Intel Core i7 CPU @3.7 GHz and nVidia Geforce GTX 2080Ti GPU. We captured live facial performance of subjects using a Logitech C922x Pro webcam which is capable of recording at 720P resolution and 60 fps. Our system ran at 28 fps for 720P video and 50 fps for 800x600 video. The speedup from the GPU solver is due to fewer levels in the hierarchy for lower resolution input video. The final frame rate is determined by the slower side: GPU in the case of 720P video and CPU in the case of 800x600 video. All the results presented in the paper and supplemental demo video were recorded at 720p for better visual quality.

Table 1: Runtime statistics for our method and timing comparison with [Cao et al. 2015] and [Guo et al. 2018]. Note that the albedo is not computed per frame in our method.

		Ours		
	Albedo	Lighting	Displacement	
1280x720	112 ms	1 ms	35 ms	28 fps
800x600	98 ms	1 ms	16 ms	50 fps
		[Cao et al. 2015]		
		Global Tracker	Local Detail	
648x860		32 ms	23 ms	18 fps
		[Guo et al. 2018]		
		CoarseNet	FineNet	
256x256		5 ms	15 ms	50 fps

5.2 Evaluation

We demonstrate the accuracy and effectiveness of our method in Figure 9, where fine-scale facial geometries are captured for subjects with various skin colors, head poses, expressions, and wrinkles. We show that our method is capable of capturing fine-scale skin details from shading changes for live camera video streams as well as legacy video footages.

Quantitative Evaluation. We first quantitatively evaluated our method by comparing the input video (i.e., images) with the synthesized textured face using estimated head pose, geometry, illumination, and albedo. As shown in Figure 10, the photometric error is relatively low except at the corner of the forehead where the highlight breaks the Lambertian surface assumption. We then evaluated our method on the binocular FaceCap dataset [Valgaerts et al. 2012], where we only use the image sequence from one camera as input. We registered our mesh with the ground-truth mesh from [Valgaerts et al. 2012] and computed the point-to-mesh distance. As shown in Figure 11, our method achieved the average error = 1.96 mm and the standard deviation = 1.35 mm, which is more accurate than the state-of-the-art, CNN-based, real-time method [Guo et al. 2018] that had the average error = 2.08 mm and the standard deviation = 1.63 mm.

Comparisons with Off-line Methods. We compared our method with the state-of-the-art off-line methods that reconstruct faces from RGB video. Beeler et al. [2011] capture high quality facial performance with multiple synchronized cameras. This hardware setup prevents their method from processing legacy video footages (e.g., Youtube video). Moreover, their method requires the manual selection of an anchor frame while our method runs in a fully automatic manner. The work of [Garrido et al. 2013] also lacks the capability to process legacy video as it relies on a high quality face scan of the subject for blendshape construction. Besides, offline methods typically involve iterative refinements across the whole sequence, thereby are unsuitable for on-line tracking. As shown in Figure 12, our method produced the results of close quality to them but in real-time, without any preprocessing for the subject or forward/backward information of input video.

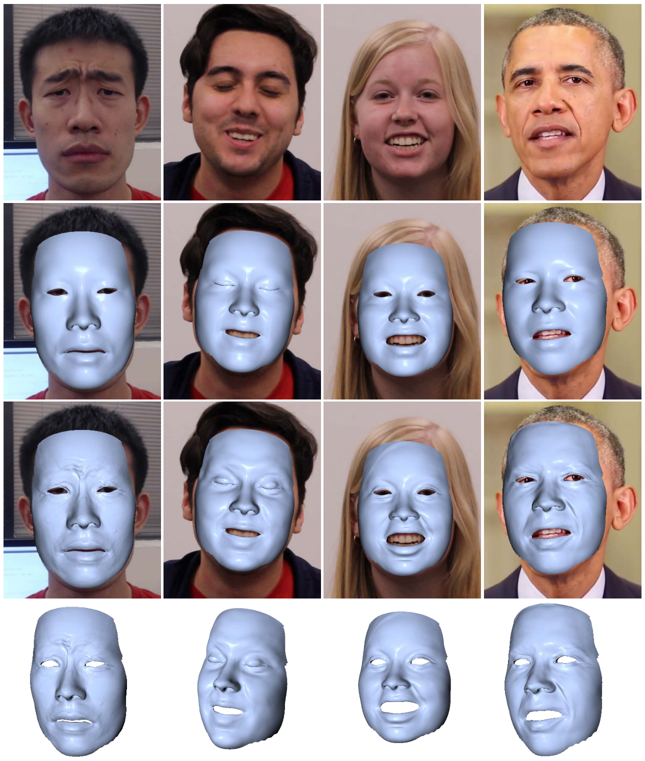


Figure 9: Our method captures coarse-scale (the second row) facial performance as well as fine-scale (the third and fourth rows) details on various identities, expressions, and head poses without any preprocessing or manual corrections. ©public domain.

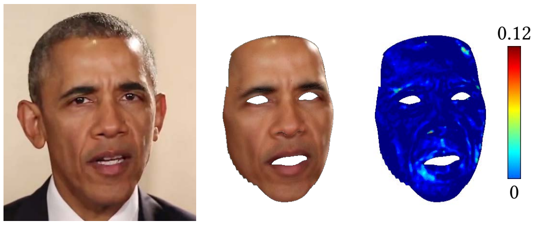
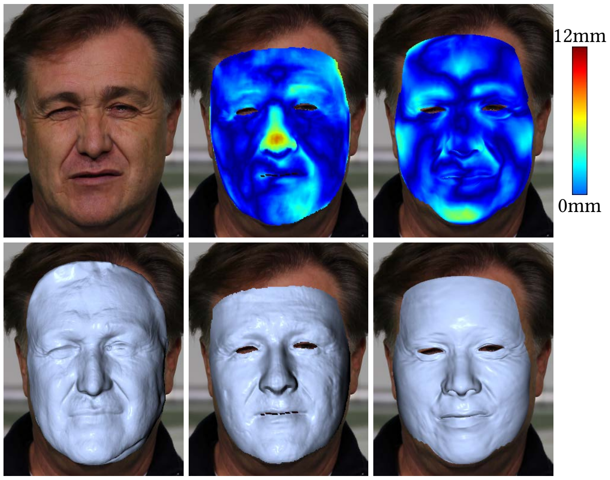


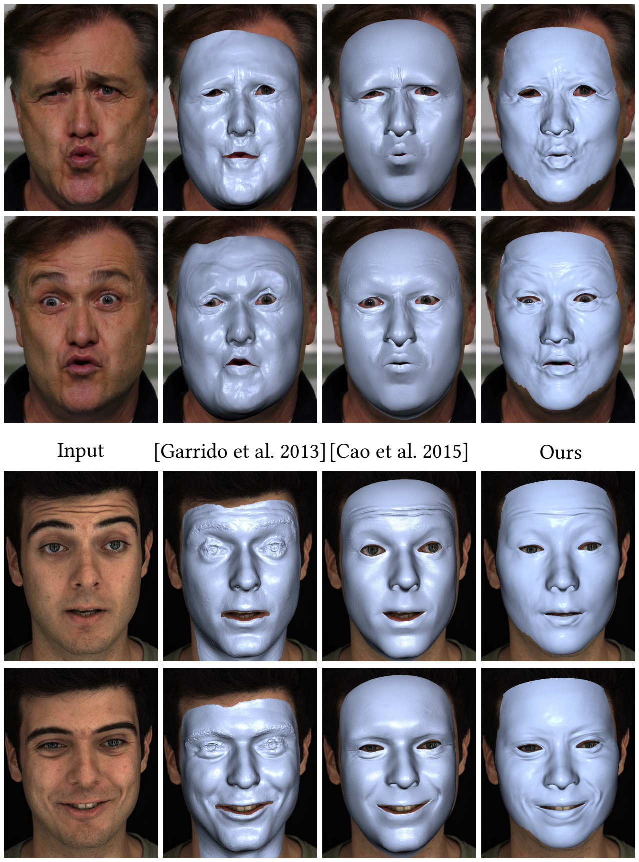
Figure 10: The photometric accuracy of our method (from left to right): the input frame, our rendered face, and heat map showing photometric errors. ©public domain.

Comparisons with Real-time Methods. Most existing real-time methods [Cao et al. 2014a, 2013; Thies et al. 2016] target on capturing coarse-scale facial expression and head motion from RGB video. Recently, Cao et al. [2015] proposed to extend the coarse global tracker with predicted local details. Their approach trains local regressors by learning correlations between image patches and surface details from a database of high-quality face scans. While plausible displacements are combined to the coarse mesh, their augmented mesh is not geometrically correct but only a close approximation of the true surface. Furthermore, medium-scale details



[Valgaerts et al. 2012] [Guo et al. 2018] Ours

Figure 11: Comparison between our method and the CNN-based method [Guo et al. 2018]. From left to right: the input video frame and ground truth mesh from the binocular method [Valgaerts et al. 2012], the result by the CNN-based method, and the result by our method.



Input [Garrido et al. 2013] [Cao et al. 2015] Ours

Input [Beeler et al. 2011] [Cao et al. 2015] Ours

Figure 12: Compared to the offline monocular method [Garrido et al. 2013] and a multi-view based method [Beeler et al. 2011], our method produces similar results. Compared to the real-time method of [Cao et al. 2015], our method excels in capturing large-scale deformations, such as sunken cheeks on the top two rows.

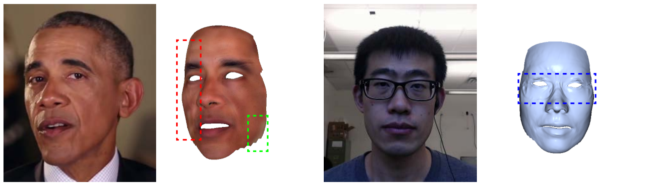


Figure 13: Limitations. Specular highlights (red) and occlusions (blue) cause artifacts. Insufficient head orientations during initialization leads to incomplete albedo (green). ©public domain.

are only added to locations where image patches are detected as wrinkles, while other regions are left with surface skins learned from the database. As a result, large scale deformations such as sunken cheeks (top of Figure 12) are often absent in their results. Similarly, the CNN-based real-time method by Guo et al. [2018] produces an approximated but noisy surface (Figure 11). By contrast, our method can produce truly reconstructed geometric details by minimizing the shading energy over the whole face. Therefore, our method is able to capture more accurate facial details as presented in Figures 11 and 12. Compared to the above two real-time methods, in terms of running time, our methods achieves highest frame rate while at higher resolutions (see Table 1).

Limitations. Our current approach relies on the detection of a sparse set of 2D facial landmarks. Inaccurate detections could lead to incorrect head poses or subject identities. Since we assume Lambertian reflectance for the face surface, specular highlights, unsmooth illumination, and cast shadows could incur artifacts. Similarly, occlusions such as hair and glasses might be misinterpreted as geometric changes. The albedo texture is recovered within the first several seconds. Therefore, as much as possible face orientations are encouraged during this stage for a complete albedo texture recovery. Figure 13 shows some failure cases for specular highlights, occlusions, and incomplete albedo texture on a novel head pose.

6 CONCLUSION

We present a real-time, automatic, geometry-based method for capturing fine-scale facial performance from monocular RGB video. Our method reconstructs large-scale head poses and expressions as well as fine-scale wrinkles, lighting, and albedo in parallel and in real-time. A novel hierarchical reconstruction method is also introduced to robustly solve the shape-from-shading surface refinements of human faces. We demonstrate that our approach can produce results close to off-line methods and better than previous real-time methods.

As the future work, we would like to enhance the coarse face tracking algorithm by combining the dense correspondences of RGB values in the image space by introducing a face texture prior. However, this will bring extra computation cost on the GPU and breaks the current CPU-GPU parallel structure, since the photometric error will be transferred back to CPU for tracking improvements. Therefore, a more sophisticated solver is demanded for this purpose. We are also interested in updating the albedo texture with unseen pixels in novel head pose or inferring a complete albedo texture using deep learning techniques at initialization.

ACKNOWLEDGEMENTS

We would like to thank Dr. Fuhao Shi for numerous helps and insightful discussion on face reconstruction and all the participants in our data acquisition. We also would like to thank Chen Cao and Yudong Guo for providing their results for comparison. This research is in part funded by NSF IIS-1524782. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the agencies.

REFERENCES

Ronen Basri and David W Jacobs. 2003. Lambertian reflectance and linear subspaces. *IEEE transactions on pattern analysis and machine intelligence* 25, 2 (2003), 218–233.

Thabo Beeler, Bernd Bickel, Paul Beardsley, Bob Sumner, and Markus Gross. 2010. High-quality Single-shot Capture of Facial Geometry. *ACM Trans. Graph.* 29, 4 (July 2010), 40:1–40:9.

Thabo Beeler, Derek Bradley, Henning Zimmer, and Markus Gross. 2012. Improved reconstruction of deforming surfaces by cancelling ambient occlusion. In *European Conference on Computer Vision*. Springer, Springer Berlin Heidelberg, Berlin, Heidelberg, 30–43.

Thabo Beeler, Fabian Hahn, Derek Bradley, Bernd Bickel, Paul Beardsley, Craig Gotsman, Robert W. Sumner, and Markus Gross. 2011. High-quality Passive Facial Performance Capture Using Anchor Frames. *ACM Trans. Graph.* 30, 4 (July 2011), 75:1–75:10.

Amit H. Bermano, Derek Bradley, Thabo Beeler, Fabio Zund, Derek Nowrouzezahrai, Ilya Baran, Olga Sorkine-Hornung, Hanspeter Pfister, Robert W. Sumner, Bernd Bickel, and Markus Gross. 2014. Facial Performance Enhancement Using Dynamic Shape Space Analysis. *ACM Trans. Graph.* 33, 2 (April 2014), 13:1–13:12.

Bernd Bickel, Mario Botsch, Roland Angst, Wojciech Matusik, Miguel Otaduy, Hanspeter Pfister, and Markus Gross. 2007. Multi-scale Capture of Facial Geometry and Motion. *ACM Trans. Graph.* 26, 3, Article 33 (July 2007).

Volker Blanz and Thomas Vetter. 1999. A Morphable Model for the Synthesis of 3D Faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM, 187–194.

Derek Bradley, Wolfgang Heidrich, Tiberiu Popa, and Alla Sheffer. 2010. High Resolution Passive Facial Performance Capture. *ACM Trans. Graph.* 29, 4 (July 2010), 41:1–41:10.

Chen Cao, Derek Bradley, Kun Zhou, and Thabo Beeler. 2015. Real-time High-fidelity Facial Performance Capture. *ACM Trans. Graph.* 34, 4 (July 2015), 46:1–46:9.

Chen Cao, Qiming Hou, and Kun Zhou. 2014a. Displaced Dynamic Expression Regression for Real-time Facial Tracking and Animation. *ACM Trans. Graph.* 33, 4 (July 2014), 43:1–43:10.

Chen Cao, Yanlin Weng, Stephen Lin, and Kun Zhou. 2013. 3D Shape Regression for Real-time Facial Animation. *ACM Trans. Graph.* 32, 4 (July 2013), 41:1–41:10.

Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. 2014b. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics* 20, 3 (2014), 413–425.

Yen-Lin Chen, Hsiang-Tao Wu, Fuhao Shi, Xin Tong, and Jinxiang Chai. 2013. Accurate and robust 3d facial capture using a single rgbd camera. In *2013 IEEE International Conference on Computer Vision*. 3615–3622.

Graham Fyffe, Andrew Jones, Oleg Alexander, Ryosuke Ichikari, and Paul Debevec. 2014. Driving High-Resolution Facial Scans with Video Performance Capture. *ACM Trans. Graph.* 34, 1 (Dec. 2014), 8:1–8:14.

Pablo Garrido, Levi Valgaert, Chenglei Wu, and Christian Theobalt. 2013. Reconstructing Detailed Dynamic Face Geometry from Monocular Video. *ACM Trans. Graph.* 32, 6 (Nov. 2013), 158:1–158:10.

Pablo Garrido, Michael Zollhöfer, Dan Casas, Levi Valgaerts, Kiran Varanasi, Patrick Pérez, and Christian Theobalt. 2016. Reconstruction of Personalized 3D Face Rigs from Monocular Video. *ACM Trans. Graph.* 35, 3 (May 2016), 28:1–28:15.

Paulo Gotardo, Jérémy Riviere, Derek Bradley, Abhijeet Ghosh, and Thabo Beeler. 2018. Practical Dynamic Facial Appearance Modeling and Acquisition. *ACM Trans. Graph.* 37, 6, Article 232 (Dec. 2018), 13 pages.

Yudong Guo, Juyong Zhang, Jianfei Cai, Boyi Jiang, and Jianmin Zheng. 2018. CNN-based Real-time Dense Face Reconstruction with Inverse-rendered Photo-realistic Face Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).

Berthold KP Horn. 1975. Obtaining shape from shading information. *The psychology of computer vision* (1975), 115–155.

Pei-Lun Hsieh, Chongyang Ma, Jihun Yu, and Hao Li. 2015. Unconstrained realtime facial performance capture. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1675–1683.

Haoda Huang, Jinxiang Chai, Xin Tong, and Hsiang-Tao Wu. 2011. Leveraging Motion Capture and 3D Scanning for High-fidelity Facial Performance Acquisition. *ACM Trans. Graph.* 30, 4 (July 2011), 74:1–74:10.

Alexandru Eugen Ichim, Sofien Bouaziz, and Mark Pauly. 2015. Dynamic 3D Avatar Creation from Hand-held Video Input. *ACM Trans. Graph.* 34, 4 (July 2015), 45:1–45:14.

Ira Kemelmacher-Shlizerman and Ronen Basri. 2011. 3D face reconstruction from a single image using a single reference face shape. *IEEE transactions on pattern analysis and machine intelligence* 33, 2 (2011), 394–405.

Oliver Klehm, Fabrice Rousselle, Marios Papas, Derek Bradley, Christophe Hery, Bernd Bickel, Wojciech Jarosz, and Thabo Beeler. 2015. Recent advances in facial appearance capture. *Computer Graphics Forum* 34, 2 (2015), 709–733.

Hao Li, Jihun Yu, Yuting Ye, and Chris Bregler. 2013. Realtime Facial Animation with On-the-fly Correctives. *ACM Trans. Graph.* 32, 4 (July 2013), 42:1–42:10.

Wan-Chun Ma, Andrew Jones, Jen-Yuan Chiang, Tim Hawkins, Sune Frederiksen, Pieter Peers, Marko Vukovic, Ming Ouhyoung, and Paul Debevec. 2008. Facial Performance Synthesis Using Deformation-driven Polynomial Displacement Maps. *ACM Trans. Graph.* 27, 5 (Dec. 2008), 121:1–121:10.

M. Sela, E. Richardson, and R. Kimmel. 2017. Unrestricted Facial Geometry Reconstruction Using Image-to-Image Translation. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 1585–1594.

Fuhao Shi, Hsiang-Tao Wu, Xin Tong, and Jinxiang Chai. 2014. Automatic Acquisition of High-fidelity Facial Performances Using Monocular Videos. *ACM Trans. Graph.* 33, 6 (Nov. 2014), 222:1–222:13.

Supasorn Suwajanakorn, Ira Kemelmacher-Shlizerman, and Steven M. Seitz. 2014. Total Moving Face Reconstruction. In *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer International Publishing, Cham, 796–812.

Supasorn Suwajanakorn, Steven M Seitz, and Ira Kemelmacher-Shlizerman. 2015. What makes tom hanks look like tom hanks. In *2015 IEEE International Conference on Computer Vision (ICCV)*. 3952–3960.

Ayush Tewari, Michael Zollhöfer, Pablo Garrido, Florian Bernard, Hyeongwoo Kim, Patrick Pérez, and Christian Theobalt. 2018. Self-supervised Multi-level Face Model Learning for Monocular Reconstruction at over 250 Hz. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ayush Tewari, Michael Zollöfer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Theobalt Christian. 2017. MoFA: Model-based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction. In *The IEEE International Conference on Computer Vision (ICCV)*.

Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. 2015. Real-time Expression Transfer for Facial Reenactment. *ACM Trans. Graph.* 34, 6 (Oct. 2015), 183:1–183:14.

Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. 2016. Face2face: Real-time face capture and reenactment of rgb videos. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2387–2395.

Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Niessner. 2018a. FaceVR: Real-Time Gaze-Aware Facial Reenactment in Virtual Reality. *ACM Trans. Graph.* 37, 2, Article 25 (June 2018), 15 pages.

Justus Thies, Michael Zollhöfer, Christian Theobalt, Marc Stamminger, and Matthias Niessner. 2018b. Headon: Real-time Reenactment of Human Portrait Videos. *ACM Trans. Graph.* 37, 4, Article 164 (July 2018), 13 pages.

Levi Valgaerts, Chenglei Wu, Andrés Bruhn, Hans-Peter Seidel, and Christian Theobalt. 2012. Lightweight Binocular Facial Performance Capture Under Uncontrolled Lighting. *ACM Trans. Graph.* 31, 6 (Nov. 2012), 187:1–187:11.

Luiz Velho and Denis Zorin. 2001. 4-8 Subdivision. *Comput. Aided Geom. Des.* 18, 5 (June 2001), 397–427.

Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popović. 2005. Face Transfer with Multilinear Models. *ACM Trans. Graph.* 24, 3 (July 2005), 426–433.

Daniel Vlasic, Pieter Peers, Ilya Baran, Paul Debevec, Jovan Popović, Szymon Rusinkiewicz, and Wojciech Matusik. 2009. Dynamic Shape Capture Using Multi-view Photometric Stereo. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 174:1–174:11.

Congyi Wang, Fuhao Shi, Shihong Xia, and Jinxiang Chai. 2016. Realtime 3D Eye Gaze Animation Using a Single RGB Camera. *ACM Trans. Graph.* 35, 4 (July 2016), 118:1–118:14.

Thibaut Weise, Sofien Bouaziz, Hao Li, and Mark Pauly. 2011. Realtime Performance-based Facial Animation. *ACM Trans. Graph.* 30, 4 (July 2011), 77:1–77:10.

Thibaut Weise, Hao Li, Luc Van Gool, and Mark Pauly. 2009. Face/Off: Live Facial Puppetry. In *SCA’09*. ACM, New York, NY, USA, 7–16.

Lance Williams. 2006. Performance-driven Facial Animation. In *ACM SIGGRAPH 2006 Courses (SIGGRAPH ’06)*. ACM, New York, NY, USA, Article 16.

Chenglei Wu, Carsten Stoll, Levi Valgaerts, and Christian Theobalt. 2013. On-set Performance Capture of Multiple Actors with a Stereo Camera. *ACM Trans. Graph.* 32, 6 (Nov. 2013), 161:1–161:11.

Chenglei Wu, Kiran Varanasi, Yebin Liu, Hans-Peter Seidel, and Christian Theobalt. 2011. Shading-based dynamic shape refinement from multi-view video under general illumination. In *ICCV’11*. ACM, New York, NY, USA, 1108–1115.

Chenglei Wu, Michael Zollhöfer, Matthias Nießner, Marc Stamminger, Shahram Izadi, and Christian Theobalt. 2014. Real-time Shading-based Refinement for Consumer Depth Cameras. *ACM Trans. Graph.* 33, 6 (Nov. 2014), 200:1–200:10.

Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. 2004. Spacetime Faces: High Resolution Capture for Modeling and Animation. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 548–558.

Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, and Marc Stamminger. 2014. Real-time Non-rigid Reconstruction Using an RGB-D Camera. *ACM Trans. Graph.* 33, 4 (July 2014), 156:1–156:12.

Michael Zollhöfer, Justus Thies, Pablo Garrido, Derek Bradley, Thabo Beeler, Patrick PÃľrez, Marc Stamminger, Matthias NieÃ§ner, and Christian Theobalt. 2018. State of the Art on Monocular 3D Face Reconstruction, Tracking, and Applications. *Computer Graphics Forum* 37, 2 (2018), 523–550.