Unconstrained Realtime Facial Performance Capture

Pei-Lun Hsieh[*] Chongyang Ma[*] Jihun Yu[†] Hao Li[*]
[*] University of Southern California [†] Industrial Light & Magic

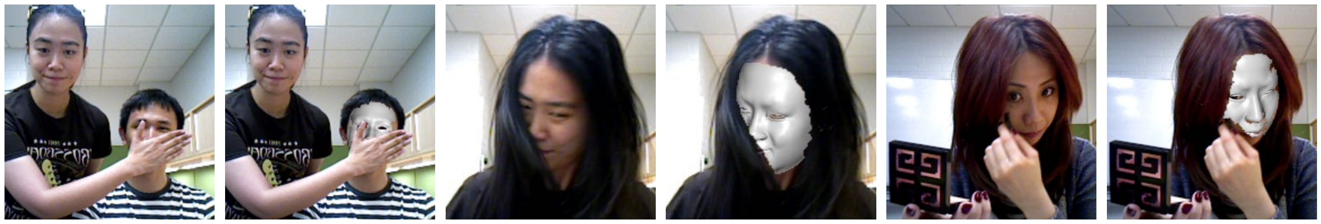
Figure 1: Calibration-free realtime facial performance capture on highly occluded subjects using an RGB-D sensor.

Abstract

We introduce a realtime facial tracking system specifically designed for performance capture in unconstrained settings using a consumer-level RGB-D sensor. Our framework provides uninterrupted 3D facial tracking, even in the presence of extreme occlusions such as those caused by hair, hand-to-face gestures, and wearable accessories. Anyone's face can be instantly tracked and the users can be switched without an extra calibration step. During tracking, we explicitly segment face regions from any occluding parts by detecting outliers in the shape and appearance input using an exponentially smoothed and user-adaptive tracking model as prior. Our face segmentation combines depth and RGB input data and is also robust against illumination changes. To enable continuous and reliable facial feature tracking in the color channels, we synthesize plausible face textures in the occluded regions. Our tracking model is personalized on-the-fly by progressively refining the user's identity, expressions, and texture with reliable samples and temporal filtering. We demonstrate robust and high-fidelity facial tracking on a wide range of subjects with highly incomplete and largely occluded data. Our system works in everyday environments and is fully unobtrusive to the user, impacting consumer AR applications and surveillance.

1. Introduction

Facial performance capture is well-established in the film and game industries for efficient and realistic animation production. While professional studios tend to rely on sophisticated solutions, *realtime* and *markerless* tracking technologies using *lightweight* monocular sensors (video or depth cameras) are becoming increasingly popular, due to their ease of adoption, cost, and deployability.

In production, the capture process is typically *constrained* for optimal performance: face visibility is maximized; the environment is well lit; and an optimal facial tracking model is built before tracking. Unconstrained facial performance capture, on the other hand, has the potential to impact surveillance, recognition, and numerous applications in the consumer space, such as personalized games, make-up apps, and video chats with virtual avatars. In these unconstrained scenarios, new challenges arise: (1) occlusions caused by accessories, hair, and involuntary hand-to-face gesticulations challenge the face segmentation problem; (2) a facial tracking model needs to be constructed on-the-fly to enable instantaneous tracking and user switching for unobtrusive performance capture.

While recent advances have shown promising results in facilitating unconstrained facial tracking with data-driven methods, they do not ensure uninterrupted tracking in the presence of large and unexpected occlusions. Driven by the growing availability of consumer-level realtime depth sensors, we leverage the combination of reliable depth data and RGB video and present a realtime facial capture system that maximizes uninterrupted performance capture in the wild. It is designed to handle large occlusion and smoothly varying but uncontrolled illumination changes. Our system also allows instant user switching without any facial calibration (Figure 1).

Our approach unifies facial tracking, segmentation, and tracking model personalization in both depth and RGB channels. We detect dynamic occlusions caused by temporal shape and texture variations using an outlier voting scheme in superpixel space. As recently demonstrated by Li *et al*. [31], the combination of sparse 2D facial features (e.g., eyes, eyebrows, and mouth) with dense depth maps are particularly effective in improving tracking fidelity. However, because facial landmark detection becomes significantly less reliable when the face is occluded, we synthesize plausible face textures right after our face segmentation step. Concurrent to the facial tracking thread, we progressively personalize our tracking model to the current user as more data is being collected. In each frame, we simultaneously solve for the user's identity (neutral pose) and expression. In summary, we make the following contributions:

- A framework that unifies tracking, segmentation, and model personalization, designed to provide unobtrusive and uninterrupted realtime 3D facial tracking.

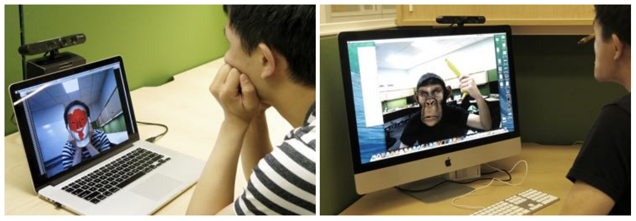

Figure 2: Our capture setup for tracking and retargeting.

- A user-adaptive face segmentation method that combines RGB and depth input for robustness w.r.t. occlusions and illumination changes. We synthesize face textures in occluded regions from the tracked model to improve the reliability of facial landmark detection.

- An on-the-fly textured blendshape personalization algorithm based on exponential temporal smoothing that selectively improves the identity and each expression depending on the tracked pose.

2. Related Work

Performance-driven facial animation has been introduced to reduce the amount of manual work involved in animation production. Pighin and Lewis present an overview of the most fundamental techniques in [32]. Production methods typically aim for tracking fidelity rather than deployability and rely on a sophisticated capture settings and carefully crafted tracking models [25, 42, 21, 28, 40, 2].

For surveillance and facial recognition, monocular 2D input is preferred due to their availability and the tracking is often designed to operate in uncontrolled environments. Methods based on parametric models have been introduced a decade ago [29, 7, 3, 19, 17] and also used for performance-driven facial animation [33, 14, 12]. Data-driven algorithms such as the popular active appearance models (AAM) [15] and constrained local models (CLM) [16] have been introduced to enable realtime tracking of sparse 2D facial features. More recently, advances in realtime 2D tracking based on landmark prediction [37] or supervised descent method [41] (distributed as CMU's IntraFace) do not involve user-specific training and have shown impressive accuracy compared to previous methods.

With the democratization of consumer-grade depth sensors, realtime facial performance capture techniques that were originally based on sophisticated structured light systems [42, 40] were quickly deployed in non-professional environments. The data-driven method of Weise *et al.* [39] uses a motion prior database to handle noise and the low-resolution depth maps from the Kinect sensor. For improved fidelity, techniques that combine depth input data with sparse facial features were introduced [31, 11, 13, 9]. To improve accessibility with less input training, an example-based facial rigging method was introduced by Li *et al.* [30]. The method of [31] builds a single neutral model before tracking and trains PCA-based correctives for the expressions during tracking with samples obtained from per-vertex Laplacian deformations. Bouaziz *et al.* [6] introduce a completely calibration-free system by modeling the full blendshape model during tracking. Even though previous frames are aggregated using exponential decay, uncontrolled variation in individual expressions can still occur during tracking. Since they optimize for all mesh vertices, the localities of the expressions are not preserved.

With sufficient data-driven priors or computation, researchers have recently demonstrated comparable 3D tracking results using purely 2D video cameras. A combination of dense flow and shape-from-shading has been proposed by Garrido *et al.* [22] for highly accurate performance capture, but involves costly computations. The 3D shape regression framework of [10] exhibits impressive 3D tracking fidelity but requires an extra offline calibration for each user. The recent work of [9] eliminates the need for tedious user-specific training using runtime regression, but requires sufficient visibility of the face for tracking and does not capture facial details as faithfully as depth-based systems.

Face segmentation from video alone is particularly challenging because of uncontrolled lighting and the wide variations of facial appearance (skin color, facial hair, make-up, scars, etc.). Handling occlusions is an ongoing thread in facial recognition and some recent developments suggest the modeling of partial occlusions directly in the training sets [26]. Data-driven approaches with occluded training samples have also been explored for 2D facial feature tracking such as the AAM framework of Gross *et al.* [24]. The state-of-the-art methods are based on discriminative trained deformable parts [34, 23] and cascade of regressors trained with occlusion data [8].

3. Overview

Our architecture is shown in Figure 3 with input data obtained from a PrimeSense Carmine 1.09 sensor. There are three main components in our system: realtime facial tracking (blue), face segmentation with occlusion completion (purple), and tracking model personalization (pink) which runs concurrently to the tracking and segmentation thread.

Tracking. Facial tracking is achieved by fitting a textured 3D tracking model to every captured RGB-D frame. First the rigid motion is estimated between the input data and the tracking model obtained in the previous frame. A user-adaptive tracking model is then used to solve for the linear blendshape expressions. Next a Laplacian deformation is applied to the tracked blendshape for accurate per-vertex displacements in the final output. Similar to the tracking method of [31] and [13], we use sparse facial features detected in the RGB channels to improve the facial tracking fidelity and to better handle fast motions in xy-directions. We use 36 out of 49 landmarks (eyebrows, eye and mouth contours) obtained from the supervised descent method of [41]. The tracked model can be used for augmented reality applications such as retargeting. To do so, we simply transfer the computed blendshape coefficients of the subject's face to a target blendshape model (Figure 3 bottom right).

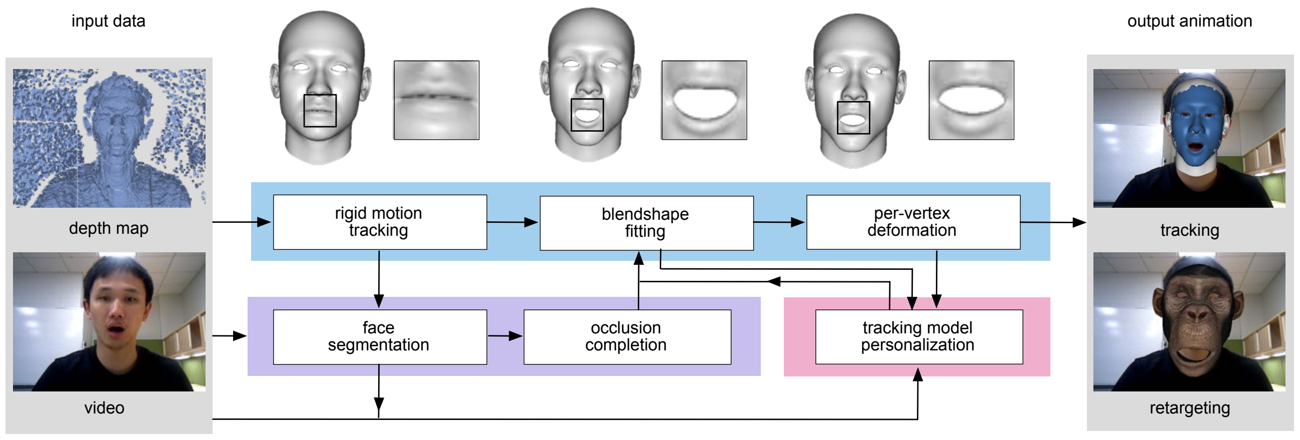

Figure 3: System overview: we unify facial tracking (blue), face segmentation (purple), and model personalization (pink).

Segmentation. The facial expressions should be solved with constraints that are defined in unoccluded regions and visible to the sensor. We therefore compute a binary segmentation map for every frame by labeling each pixel as face region or occlusion in the UV map of the tracking model. We model occlusions as outliers in the input data using the vertex positions and texture of the exponentially smoothed tracking model as reference. While only unoccluded regions are used for tracking, we fill the occluded ones with textures that are aggregated on the tracked face model from the previous frames. By synthesizing the facial features behind occlusions, landmark detection becomes significantly more reliable.

Personalization. Our tracking model is initialized with a generic blendshape model (statistical mean and 28 generic FACS-based expressions [18]) which is adapted to the user during tracking. Every time the template personalization updates its shape and appearance, the latest one is retrieved and used for tracking. For every input frame, the blendshape coefficients computed by the facial tracking are used to solve for the shape of the user's identity using a linear PCA model. Next, the mesh vertices of the expression shapes are refined to match the captured subject. To account for the entire history of personalized tracking models, we recursively aggregate the new shapes and recorded texture to the previous ones via exponentially weighted moving average. Only those expressions are solved, if the currently tracked model is closer to the corresponding expression than all previous observations. The tracking model effectively improves over time and uncontrolled shape variations of the tracking model can be significantly reduced.

4. Realtime Facial Tracking

We adopt a similar tracking pipeline as [31] which combines linear blendshape tracking with per-vertex Laplacian deformations for improved accuracy. The tracking template is simultaneously matched to the input depth maps and sparse facial features. Prior to tracking, we first detect the face position using the facial feature corresponding to the tip of the nose and crop a region of interest obtained from [41] within a radius of 50 pixels. We re-detect the face if the rigid motion computation does not converge or fails a *penetration test*, i.e., when 25% of the depth map correspondences are more than 5 mm behind the tracking model. The penetration test ensures that occluding objects are always in front of the face.

Our tracking model consists of (1) rigid motion estimation, (2) linear blendshape fitting, and (3) per-vertex Laplacian deformation as illustrated in Figure 3. Our base tracking model is driven by a linear blendshape model with vertex $\mathbf{v}(\mathbf{x}) = \mathbf{b}_0^t + \mathbf{B}^t\mathbf{x}$, where $\mathbf{b}_0^t$ is the personalized mesh of the neutral pose, the columns of $\mathbf{B}^t$ the mesh of the personalized expressions, and $\mathbf{x} \in [0,1]^N$ the $N = 28$ blendshape coefficients. The personalized meshes $\mathbf{b}_0^t$ and $\mathbf{B}^t$ are retrieved at frame t whenever an update occurs.

Rigid Motion Estimation. We begin tracking by solving for the global rigid motion between the tracking model obtained in the previous frame and the current depth map using the iterative closest point (ICP) algorithm [36], which is based on point-to-plane constraints on the input depth map and point-to-point constraints on the 2D facial features. We prune correspondences that are further away than 5mm and normals between source and target points that are larger than 30 degrees. We allow 50 iterations of ICP, but the estimation typically converges in less than 5 iterations.

Linear Blendshape Fitting. To fit the blendshape to the current frame, we alternate between per-vertex correspondence computation and blendshape coefficients solving. We use the point-to-plane fitting term on the depth map:

$$c_i^S(\mathbf{x}) = \alpha_i^S \left(\mathbf{n}_i^\top(\mathbf{v}_i(\mathbf{x}) - \bar{\mathbf{v}}_i)\right)^2 , \qquad (1)$$

where $\mathbf{v}_i$ is the i-th vertex of the mesh, $\bar{\mathbf{v}}_i$ is the projection of $\mathbf{v}_i$ to the depth map and $\mathbf{n}_i$ the surface normals of $\bar{\mathbf{v}}_i$. A binary visibility term $\alpha_i^S \in \{0,1\}$ is assigned to every $\mathbf{v}_i$, and its value is obtained by sampling the segmentation map computed in Section 5.

We also use the point-to-point fitting term on the 2D facial features:

$$c_j^F(\mathbf{x}) = \alpha_j^F \|\pi(\mathbf{v}_j) - \mathbf{u}_j\|_2^2 \,, \tag{2}$$

where $\mathbf{u}_j$ is the position of a tracked 2D facial feature, $\pi(\mathbf{v}_j)$ its corresponding mesh vertices projected into camera space, and α_j^F its binary visibility term, which is also obtained from the same segmentation map as before.

We solve for the coefficients $\mathbf{x}$ of our linear tracking model by minimizing the total energy term using a fast iterative projection method [38] in 3 iterations:

$$\min_{\mathbf{x}} \sum_i c_i^S(\mathbf{x}) + w \sum_j c_j^F(\mathbf{x}) \,,$$

where $w = 5 \cdot 10^{-5}$ is the weight of the facial feature term and the elements of $\mathbf{x}$ are bounded between 0 and 1.

Per-Vertex Deformation. For the final tracking output we perform one step of fast Laplacian deformation $\tilde{\mathbf{v}}_i = \mathbf{v}_i + \Delta\tilde{\mathbf{v}}_i$ as proposed by [31] solving for the per-vertex displacements $\Delta\tilde{\mathbf{v}}_i$. We first use the point-to-point fitting term on the depth map:

$$c_i^P(\Delta\tilde{\mathbf{v}}_i) = \alpha_i^P \|\tilde{\mathbf{v}}_i - \bar{\mathbf{v}}_i\|_2^2 \,, \tag{3}$$

where $\bar{\mathbf{v}}_i$ is again the projection of $\mathbf{v}_i$ to the depth map and α_i^P its visibility term.

Similar to Equation 2, we derive the point-to-point fitting term on 2D facial features for Laplacian deformation:

$$c_j^W(\Delta\tilde{\mathbf{v}}_j) = \alpha_j^W \|\pi(\tilde{\mathbf{v}}_j) - \mathbf{u}_j\|_2^2 \,. \tag{4}$$

We minimize the total energy term:

$$\min_{\Delta\tilde{\mathbf{v}}_i} \sum_i c_i^P(\Delta\tilde{\mathbf{v}}_i) + w_1 \sum_j c_j^W(\Delta\tilde{\mathbf{v}}_j) + w_2 \Delta\tilde{\mathbf{v}}^\top \mathbf{L}(\mathbf{b}_0^t)\Delta\tilde{\mathbf{v}} \,,$$

where $w_1 = 10$ is the weight for the facial feature term, $w_2 = 100$ the weight for the Laplacian smoothing term, $\tilde{\mathbf{v}} = [\tilde{\mathbf{v}}_0, \ldots, \tilde{\mathbf{v}}_{V-1}]^\top$ with V the number of vertices, and $\mathbf{L}(\mathbf{b}_0^t)$ the Laplacian regularization matrix with cotangent weights w.r.t. the current neutral mesh [5]. We solve for the displacement $\Delta\tilde{\mathbf{v}}_i$ using the sparse linear system LDLT solver from the C++ Eigen Library. Because $\mathbf{b}_0^t$ is progressively updated by the tracking model personalization (see Section 6), we factorize of the sparse linear system in a separate thread to ensure realtime performance.

5. Face Segmentation

We explicitly segment the face during tracking for effective occlusion handling. We use the incoming RGB-D frame and the current tracking model to produce a binary segmentation map $\mathbf{M}$ in texture space. The segmentation map represents the visibility of the tracked face (Section 4) and the region of interest for the tracking model personalization (Section 6). Because of the limited resolution of our RGB-D input, we represent $\mathbf{M}$ as a small 100×100 image.

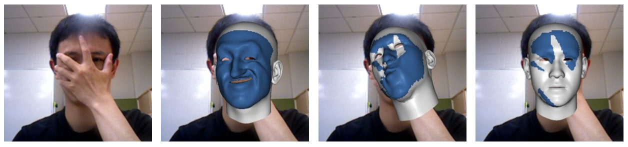

Figure 4: Segmentation with depth and penetration tests.

We perform a binary classification on $\mathbf{M}$ and assign to a pixel $m_i = 1$ if it is part of the face and $m_i = 0$ if it is either occluded or invisible to the camera. Our general approach for detecting occluded pixels consists of comparing dynamic variations, or *outliers*, between captured input frame and the tracking model using all RGB-D channels. Because our tracking model and its texture are averaged over all previous frames using exponential smoothing, they form an robust prior for outlier detection. Exponential smoothing also allows the tracking model to rapidly adapt its shape and appearance to a new user while being resistant to smooth lighting variations of the environment.

Depth Input. Like other 3D sensor-based methods [39, 6, 31], we use the depth map for rough occlusion detection. The tracked model of the previous frame is used right after rigid motion estimation, and assigns $m_i = 0$ if its L_2 distance to the depth map is larger than a threshold $\sigma_D = 5$mm and $m_i = 1$ otherwise. If σ_D is too small, input expressions with vertex motions in z-direction may not be captured correctly, and if σ_D is too large, it becomes difficult to discern between the face and occlusions. While this segmentation is invariant to appearance and illumination variations, it is challenged by the noise and limited resolution of current consumer-grade sensors. As proposed in Section 4, we use an additional penetration test to ensure that occlusions are always in front of the face to improve the reliability of the segmentation (see Figure 4).

Video Input. Only inliers from the depth input are considered for color segmentation. While depth data is very effective for face segmentation with large geometric occlusions (e.g., hand-to-face gestures), thin structures such as hair, tongue, glasses, and other accessories are generally difficult to detect from depth alone leading to failure of previous depth-based methods. Because the space of variations of occlusions is hard to model in the appearance domain [8, 34, 23], we use a history of input frames to determine the segmentation map. Similar to depth, we model occlusions in the color channel as outliers between the input color frame and the exponentially smoothed texture map that is aggregated from all previous frames.

Naïve per-pixel thresholding on $\mathbf{M}$ is prone to errors because of noise in the input video, specularity on faces, and complex lighting variations. We therefore enforce smooth spatial and color coherence in our segmentation for additional robustness. We first convert our RGB input frame into CIELAB color space for a more robust treatment w.r.t.

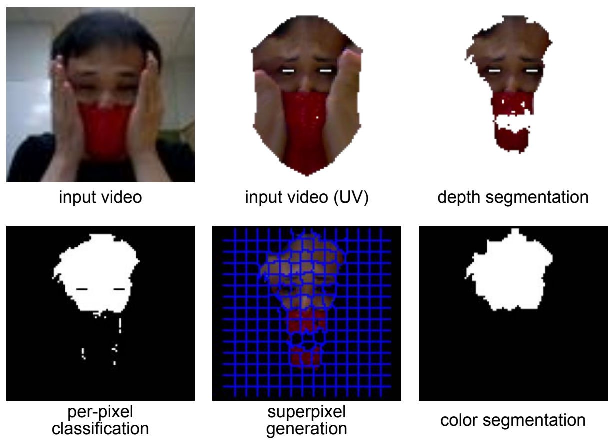

Figure 5: Video-based segmentation with superpixels.

illumination variance. We adopt the simple linear iterative clustering (SLIC) of [1] to generate *superpixels* from the current texture map of the tracking model. Superpixels are grouped pixels, or atomic regions, that are close in color and spatial proximity. They are ideally suited for efficient image segmentation since they are perceptually meaningful and adhere well to object boundaries.

SLIC is a k-means clustering algorithm with limited window search which makes it particularly fast and memory efficient. The K superpixels are each represented by a cluster center $C_k = [l_k, a_k, b_k, u_k, v_k]$ which concatenates an LAB color $[l_k, a_k, b_k]$ and a uv-position $[u_k, v_k]$. C_k is initialized on a regular grid with interval $S = \sqrt{N/K}$, where N is the number of pixels in $\mathbf{M}$. For each C_k, the algorithm repeatedly assigns to a pixel i in a $2S \times 2S$ region around C_k a label $l(i)$ if its distance is closer to C_k than any other cluster. The C_ks are then updated with the mean $[l, a, b, u, v]$ values of pixels with $l(i) = k$. We repeat this procedure 3 times and choose $k = 200$.

Since superpixels do not provide a binary segmentation directly, we combine superpixel computation with our per-pixel thresholding approach (see Figure 5). We first perform a per-pixel outlier detection and set $m_i = 0$ if the L_2 distance in the ab channels between current frame and the texture of the tracking model is above a threshold $\sigma_C = 16$ and $m_i = 1$ otherwise. We then count the number of per-pixel inliers in each cluster C_k and set all pixels with $l(i) = k$ to $m_i = 1$ if more than 50% of the pixels in each C_k are inliers and $m_i = 0$ otherwise.

Occlusion Completion. After face segmentation, we fill the occluded regions $m_i = 0$ by transferring the colors of the adaptive tracking model texture map. This step maximizes the visibility of face textures during occlusion for more accurate and reliable 2D facial landmark detection. While it is impossible to know the texture of an occluded region, we can synthesize a plausible appearance that matches the last observation (Figure 6).

Note that for longer periods of occlusions, discontinuous color transitions between the synthesized and non-occluded regions can occur due to illumination changes in the environment. While a smooth blend between the regions would improve the visual quality of the overall face texture, it does not affect the quality of the landmark detection of [41].

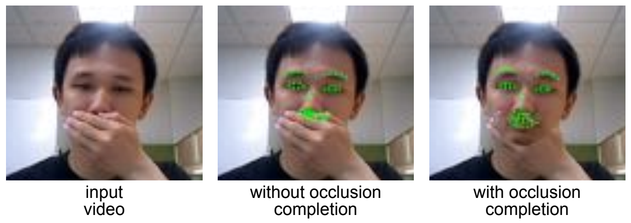

Figure 6: Occlusion completion improves the reliability of 2D facial feature tracking.

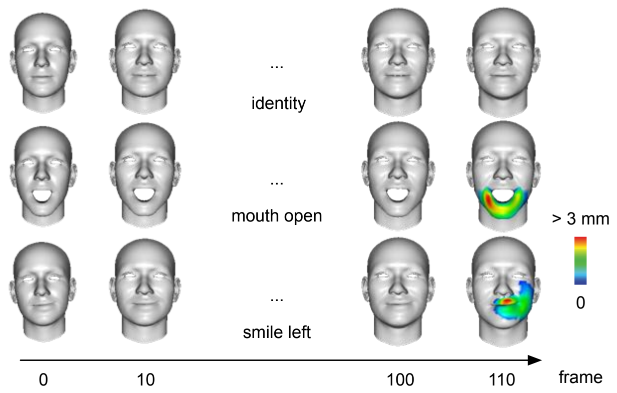

Figure 7: We continuously adapt the identity and expressions of the template model to the user during tracking.

6. Tracking Model Personalization

In parallel to the facial tracking thread, the tracking model is progressively optimized to fit to the user's shape and appearance. Because neither the identity nor the individual expressions are known in advance when a new user is being captured, we begin with a statistical mean identity $\mathbf{b}_0^0$ and 28 generic FACS-based expression $\mathbf{B}^0$, which are then personalized. As described in Section 4, our tracking is grounded on an adaptive linear blendshape model, where each mesh vertex is described by $\mathbf{v}(\mathbf{x}) = \mathbf{b}_0^t + \mathbf{B}^t\mathbf{x}$ with $\mathbf{x}$ the blendshape coefficients, $\mathbf{B}^t = [\mathbf{b}_1^t, \dots, \mathbf{b}_N^t]$ the expressions, and t the frame at which the model is personalized. The personalization stage uses the blendshape coefficients $\mathbf{x}$ computed by the blendshape fitting and then updates the tracking model which is then used in the next tracking iteration. Each personalization step consists of first solving the identity shape followed by optimizing all expressions. The solutions are then aggregated to the previously optimized shapes via exponential smoothing and a careful scheduling strategy.

Identity Optimization. Tracking model personalization first solves for the identity, which is represented by $\mathbf{b}_0^t(\mathbf{x}) = \mathbf{a}_0 + \mathbf{A}\mathbf{y}^t$, where $\mathbf{y}^t$ are the 50 unknown PCA coefficients, $\mathbf{a}_0 = \mathbf{b}_0^0$ the statistical mean shape, and the columns of $\mathbf{A}$ the PCA modes from the morphable face models described in [4]. We use the same depth-based point-to-plane terms $c_i^S(\mathbf{y}^t)$

and the facial feature-based point-to-point terms $c_i^F(\mathbf{y}^t)$ as for the blendshape fitting (see Section 4) to solve for $\mathbf{y}^t$. We also apply the same facial feature weight $w = 5 \cdot 10^{-5}$, but bound $\mathbf{y}^t$ between -2 and 2.

Expression Optimization. Once the identity shape is solved, we personalize all the expressions $\mathbf{B}$ to the user. Since the number of vertices is too large for efficient optimization, we reduce their dimensionality using spectral mesh processing as described in [27] and recently adopted for online modeling [6]. We extend the approach of [6] to preserve the locality of blendshape expressions.

Let $\mathbf{S}_i$ be the subset of vertices $\{\mathbf{v}_j\}$ for a blendshape $\mathbf{b}_i$, where the vertex displacements $||\mathbf{v}_j(\mathbf{b}_i) - \mathbf{v}_j(\mathbf{b}_0)||_2$ are larger than a given threshold $\epsilon = 0.1$. While in the original formulation of [6], spectral analysis is performed on the entire frontal face, we compute dedicated eigenvectors for each expression only considering vertices that contribute to deformations. We visualize the offset between the current frame and the generic expression blendshape as heatmap in Figure 7. We thereby avoid semantically erroneous improvements in regions where no displacements are expected. We then form the graph Laplacian matrix [27] $\mathbf{L}_i$ using $\mathbf{S}_i$ and the connectivity of $\mathbf{b}_i$ to compute the k eigenvectors $\mathbf{E}_i = [\mathbf{e}_{i1}, \ldots, \mathbf{e}_{ik}]$ with the smallest eigenvalues. The eigenvectors correspond to the frequency modes of a deformation field represented by $\mathbf{L}_i$, where the low frequencies correspond to those with small eigenvalues. We obtain a linearized deformation field per expression $\mathbf{E}_i\mathbf{z}_i$, where $\mathbf{z}_i$ are the k spectral coefficients.

With known identity shape $\mathbf{b}_0^t$ and blendshape coefficients $\mathbf{x}$, we now solve for the blendshapes using the eigenvectors $\{\mathbf{E}_i\}$. Because $\mathbf{E}_i$ are computed from $\mathbf{S}_i$ which have different numbers of vertices, we extend them to $\bar{\mathbf{E}}_i$ such that the entries with same vertex indices have the values of $\mathbf{E}_i$ and the rest are 0. Using the combined blendshape deformation fields $\bar{\mathbf{E}} = [\bar{\mathbf{E}}_1, \ldots, \bar{\mathbf{E}}_N]$ and concatenated spectral coefficients $\mathbf{z} = [\mathbf{z}_1^\top, \ldots, \mathbf{z}_N^\top]^\top$, we obtain the following linear representation:

$$\mathbf{v}(\mathbf{z}) = \mathbf{b}_0^t + (\mathbf{B}^t + \mathbf{E}\mathbf{z})\mathbf{x}.$$

Instead of using only depth maps constraints as in [6], we use the final Laplacian deformation $\tilde{\mathbf{v}}$ (see Section 4) similar to the on-the-fly corrective technique proposed by [31] and use the following point-to-point fitting term:

$$c_i^P(\mathbf{z}) = \alpha_i^P ||(\mathbf{v}_i(\mathbf{z}) - \tilde{\mathbf{v}}_i)||_2^2,$$

and a regularization term to reduce the aggregation of noise:

$$r(\mathbf{z}) = \sum_i (w_1 ||\mathbf{D}_i \mathbf{z}_i||_2^2 + w_2 ||\mathbf{z}_i||_2^2)$$

where $\mathbf{D}_j$ is a diagonal matrix with the eigenvalues of $\mathbf{e}_{jk}$ at its k-th diagonal element, $w_1 = 10^{-3}$ and $w_2 = 10^{-5}$ are the weights for the regularization terms. By minimizing the total energy:

$$\min_{\mathbf{z}} \sum_i c_i^P(\mathbf{z}) + r_i(\mathbf{z}),$$

we obtain the linearized deformation field for each expression and update the expressions $\mathbf{B}^{t+1} = \mathbf{B}^t + \mathbf{E}\mathbf{z}$.

Exponential Smoothing. Because our input data is incomplete and affected by noise, it is unlikely that each of our solutions for $\mathbf{y}^t$ and $\mathbf{B}^t$ is consistent over time. We therefore take into account all previously tracked face models by temporally averaging using an exponential decay. We simply use the exponentially weighted moving average formula [35] to aggregate the entire history of shapes $\mathbf{v}^t(\mathbf{x}) = \gamma\ \mathbf{v}^t(\mathbf{x}) + (1-\gamma)\ \mathbf{v}^{t-1}(\mathbf{x})$, where γ is the decay factor. We obtain the following update rules for the optimized identities and expressions $\mathbf{y}^t = \gamma\mathbf{y}^t + (1-\gamma)\mathbf{y}^{t-1}$ and $\mathbf{B}^t = \gamma\,\mathbf{B}^t + (1-\gamma)\,\mathbf{B}^{t-1}$.

We apply the same formula on the aggregated textures of the tracking model and only use non-occluded regions. Lower values of γ lead to more robustness but a slower personalization rate and adaptation to the appearance changes (e.g., caused by lighting variations). We use $\gamma = 0.2$ for all of our examples including the exponential smoothing of both shape and appearance.

Scheduling. Since the identity PCA is least local, affecting all the front facing vertices, we exclusively solve for $\mathbf{y}^t$ for $t < 100$ to ensure robust convergence of the tracking model personalization. We then solve for both the identity shape and expression shapes, and begin the accumulation of texture using exponential smoothing.

For the identity, we only aggregate if the current $\mathbf{x}$ is closer to the null vector $\mathbf{o}$ than previous observations. Similarly, we only optimize the expressions $\mathbf{b}_i$ with $i = 1, \ldots, N$, if $\mathbf{b}_i$ is closer to the i-th standard basis vector $\mathbf{i}_i$ which i-th entry is 1 and 0 for all the rest. In this way, we ensure that only those expressions are improved if the current tracking matches better than the previous ones.

7. Results

We show in Figures 1 and 8 challenging tracking examples (infants, different skin colors, specular lighting) with occlusions that frequently occur in everyday life (hands, hair, objects). By combining depth and video input, our face segmentation algorithm allows us to clearly discern between occlusions and face regions on the segmentation map. Our subjects are captured in fully uncontrolled lighting conditions. Figure 9 illustrates the adaptability of our tracking model personalization during user switching. We also evaluate the robustness of our system w.r.t. uncontrolled illumination changes in outdoor scenes (Figure 10) and extreme capture angles (Figure 12). In Figure 11, we challenge our face segmentation pipeline with different levels of occlusions that are synthetically generated on multiple sequences of facial animation. The artificial occlusions are rotated around the face center and their sizes increase gradually. Tracking accuracy is measured using the unoccluded model as ground truth and averaged over all sequences. Using both depth and RGB for segmentation (green) clearly outperforms when only depth is used (blue).

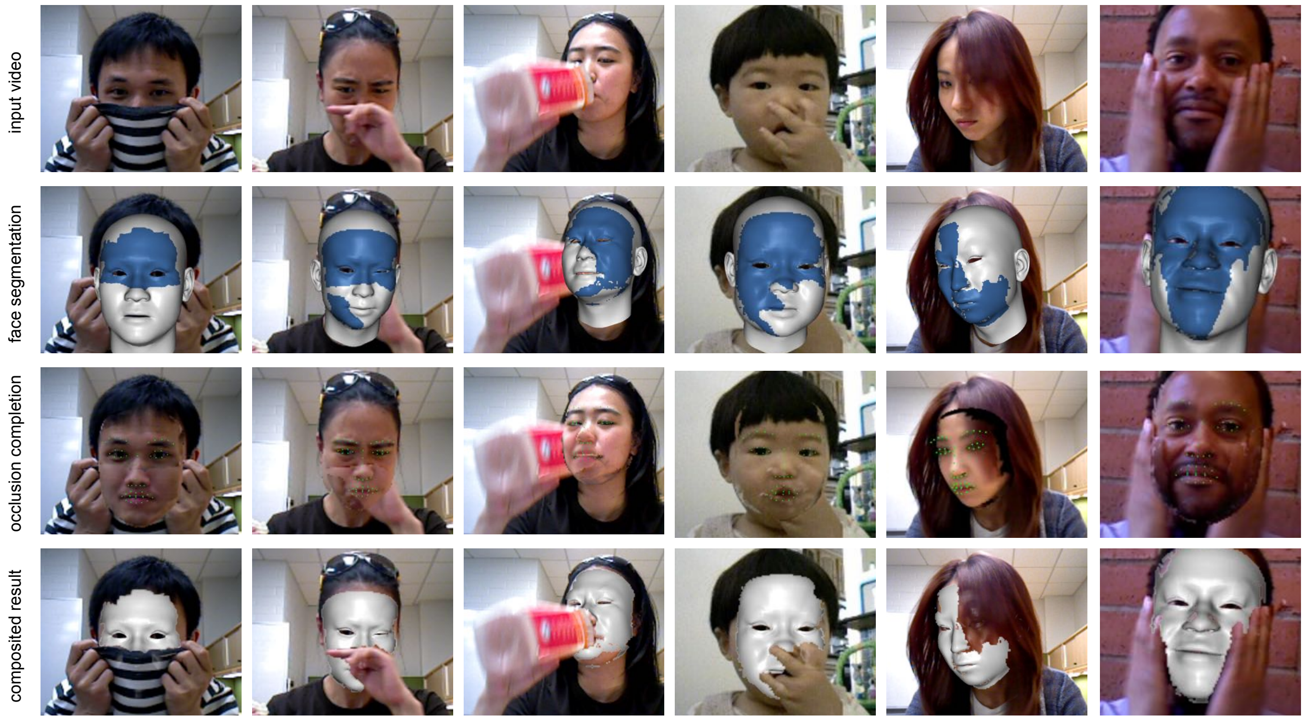

Figure 8: Our facial tracking results. We can handle challenging cases with large occlusion such as those caused by wearable accessories, hand-to-face gestures, hair and other obstacles. From top to bottom, we show the frames of input video, tracking models with inliers visualized in blue, synthesized face textures with 2D facial features, and the final composited results.

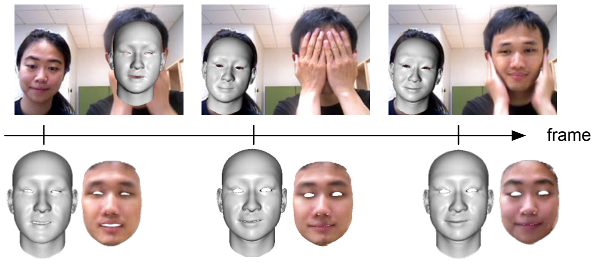

Figure 9: We can switch between users without requiring an additional calibration step. Top: input frames and tracking model; middle and bottom: shape and appearance of the tracking model.

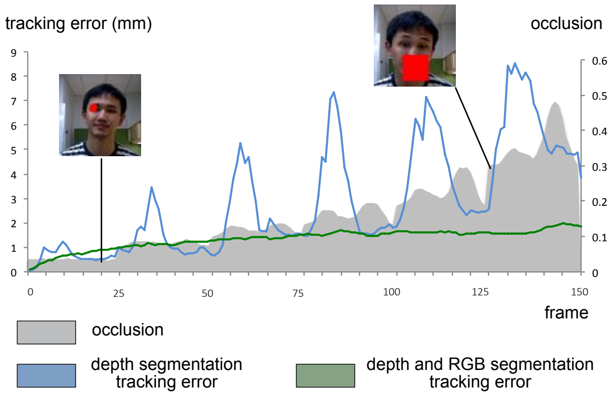

Figure 11: Impact of our segmentation on the tracking accuracy with different levels of occlusions.

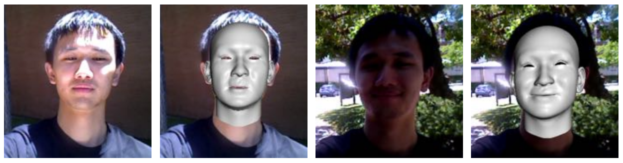

Figure 10: Our results in outdoor environments.

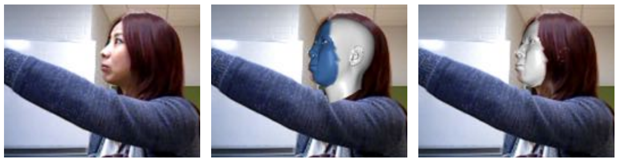

Figure 12: Our method does not break even when the subject is facing away the camera.

Comparison. We compare the performance of our occlusion handling with other state-of-the-art realtime facial capture systems (see Figure 13). While recent depth sensor-based methods [39, 31] can handle partial occlusions in the depth data if no facial features are used (row 1), we show that our explicit face segmentation can use facial features and is more effective for occlusions that are hard to detect in the depth map (row 2). The pure RGB-based technique of Cao *et al*. [9] would fail for very large occlusions (row 3)

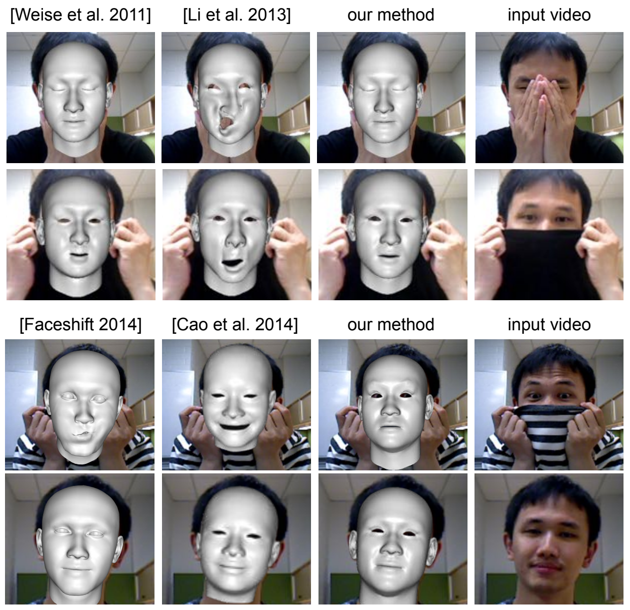

Figure 13: Comparison with other realtime techniques.

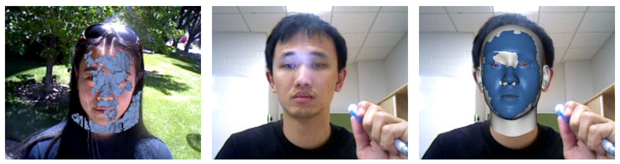

Figure 14: Limitations of our system. Left: since our RGB-D camera is based on IR illumination, our depth map can be negatively affected by direct sunlight. Middle/Left: Illuminating the directly face with a flashlight produces an outlier during face segmentation.

and the modeling is not as faithful as depth-based methods (row 4). We also compare our system with the commercial animation software Faceshift [20], which also fails for large occlusions (row 3) but produces more accurate face models (row 4) at the cost of a tedious calibration procedure.

Limitations. While our tracking model adapts to smooth appearance changes due to lighting variations, a sudden change in the lighting can still result in a wrongly detected outlier. In Figure 14, we illuminate the face with a flashlight, which results in a wrong segmentation. Even though our method is designed for uninterrupted capture, our rigid motion estimation can still fail when the face disappears completely. However, we can immediately recover the tracking by re-detecting the face in the case of such failure.

Performance. Our tracking and segmentation processes run concurrently to the tracking model personalization. The full pipeline runs at full 30 fps on a quad-core 2.3 GHz Intel Core i7 with 8GB RAM and an NVIDIA GeForce GT 650M graphics processor. While our code can be further optimized, we measure the following timings: rigid motion tracking takes 0.2 ms, blendshape fitting 1 ms, Laplacian deformation 2 ms, face segmentation 3 ms, occlusion completion 4 ms, and one step of model personalization 2 ms. The superpixel generation (SLIC) runs on the GPU and the remaining implementation is multi-threaded on the CPU.

8. Discussion

Our system demonstrates the ability to handle extremely challenging occlusions via an explicit face segmentation approach during tracking using both depth and RGB channels. By simply voting inliers in superpixel space using an appearance adaptive tracking model, our system produces clean segmentations even when the illumination changes in the environment. Unlike existing data-driven methods, our approach does not require a dedicated appearance modeling, since its construction would require a prohibitively large amount of training data to capture all the possible variations. We have also demonstrated that synthesizing face textures in the occluded regions is a crucial step to enable reliable use of landmark detection and provide accurate and continuous tracking when the face is occluded. Even though there is no solution yet for an accurate prediction of identity and expressions shapes for arbitrary users, our on-the-fly blendshape modeling solution prevents uncontrolled shape variations using localized expression optimization and blendshape coefficient monitoring. While our online generated models are close to pre-calibrated ones [20], our depth-based personalization algorithm significantly outperforms pure RGB systems [9] (see Figure 13).

Future Work. Our current face segmentation approach is purely based on a temporal prior that adapts to the user's shape and appearance. While effective in detecting dynamic occlusions, a fringe for example that is visible since the beginning can be mistakenly segmented as part of the face. When the fringe is moved away, the forehead would become an outlier. We believe that a recognition approach using prior knowledge with higher level semantics could be integrated into our framework to disambiguate these scenarios. Even though our personalized tracking model can ensure accurate tracking, the solved expression shapes may not faithfully represent those of the actor, especially when the captured face is largely incomplete. One possible future avenue consists of exploring supervised learning techniques with statistical spaces of expressions to recover expressions that truly match the user's face.

Acknowledgements We would like to thank Mikhail Smirnov for architecting the codebase; Ethan Yu, Frances Chen, Iris Wu, and Lamont Grant for being our capture models; Justin Solomon for the fruitful discussions; Kyle Olszewski and Jessica Rossky for proofreading; and the reviewers for their valuable feedbacks and suggestions. This research is based upon work supported in part by Oculus VR/Facebook, Adobe, Pelican Imaging, Panasonic, the USC Integrated Media System Center, the Google Faculty Research Award, the Office of the Director of National Intelligence (ODNI), and Intelligence Advanced Research Projects Activity (IARPA), under contract number 2014-14071600010. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Ssstrunk. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *TPAMI*, 34(11):2274 – 2282, 2012.

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

[3] M. J. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *ICCV*, pages 374–381, 1995.

[4] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH '99*, pages 187–194, 1999.

[5] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *TVCG*, 14(1):213–230, 2008.

[6] S. Bouaziz, Y. Wang, and M. Pauly. Online modeling for realtime facial animation. *ACM Trans. Graph.*, 32(4):40:1–40:10, 2013.

[7] C. Bregler and S. Omohundro. Surface learning with applications to lipreading. *Advances in neural information processing systems*, pages 43–43, 1994.

[8] X. Burgos-Artizzu, P. Perona, and P. Dollár. Robust face landmark estimation under occlusion. In *ICCV*, 2013.

[9] C. Cao, Q. Hou, and K. Zhou. Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Trans. Graph.*, 33(4):43:1–43:10, 2014.

[10] C. Cao, Y. Weng, S. Lin, and K. Zhou. 3D shape regression for real-time facial animation. *ACM Trans. Graph.*, 32(4):41:1–41:10, 2013.

[11] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. FaceWarehouse: A 3D facial expression database for visual computing. *TVCG*, 20(3):413–425, 2014.

[12] J. Chai, J. Xiao, and J. Hodgins. Vision-based control of 3D facial animation. In *SCA '03*, pages 193–206, 2003.

[13] Y.-L. Chen, H.-T. Wu, F. Shi, X. Tong, and J. Chai. Accurate and robust 3D facial capture using a single rgbd camera. In *ICCV*, pages 3615–3622, 2013.

[14] E. Chuang and C. Bregler. Performance driven facial animation using blendshape interpolation. Technical report, Stanford University, 2002.

[15] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *TPAMI*, 23(6):681–685, 2001.

[16] D. Cristinacce and T. Cootes. Automatic feature localisation with constrained local models. *Pattern Recogn.*, 41(10):3054–3067, 2008.

[17] D. Decarlo and D. Metaxas. Optical flow constraints on deformable models with applications to face tracking. *Int. J. Comput. Vision*, 38(2):99–127, 2000.

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

[19] I. Essa, S. Basu, T. Darrell, and A. Pentland. Modeling, tracking and interactive animation of faces and heads using input from video. In *Proceedings of the Computer Animation*, pages 68–79, 1996.

[20] Faceshift, 2014. `http://www.faceshift.com/`.

[21] Y. Furukawa and J. Ponce. Dense 3D motion capture for human faces. In *CVPR*, pages 1674–1681, 2009.

[22] P. Garrido, L. Valgaert, C. Wu, and C. Theobalt. Reconstructing detailed dynamic face geometry from monocular video. *ACM Trans. Graph.*, 32(6):158:1–158:10, 2013.

[23] G. Ghiasi and C. Fowlkes. Occlusion coherence: Localizing occluded faces with a hierarchical deformable part model. In *CVPR*, pages 1899–1906, 2014.

[24] R. Gross, I. Matthews, and S. Baker. Active appearance models with occlusion. *Image Vision Comput.*, 24(6):593–604, 2006.

[25] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *SIGGRAPH '98*, pages 55–66, 1998.

[26] H. Jia and A. Martinez. Support vector machines in face recognition with occlusions. In *CVPR*, pages 136–141, 2009.

[27] B. Lévy and H. R. Zhang. Spectral mesh processing. In *ACM SIGGRAPH 2010 Courses*, pages 8:1–8:312, 2010.

[28] H. Li, B. Adams, L. J. Guibas, and M. Pauly. Robust single-view geometry and motion reconstruction. *ACM Trans. Graph.*, 28(5):175:1–175:10, 2009.

[29] H. Li, P. Roivainen, and R. Forcheimer. 3-d motion estimation in model-based facial image coding. *TPAMI*, 15(6), 1993.

[30] H. Li, T. Weise, and M. Pauly. Example-based facial rigging. *ACM Trans. Graph.*, 29(4):32:1–32:6, 2010.

[31] H. Li, J. Yu, Y. Ye, and C. Bregler. Realtime facial animation with on-the-fly correctives. *ACM Trans. Graph.*, 32(4):42:1–42:10, 2013.

[32] F. Pighin and J. P. Lewis. Performance-driven facial animation. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, 2006.

[33] F. H. Pighin, R. Szeliski, and D. Salesin. Resynthesizing facial animation through 3D model-based tracking. In *ICCV*, pages 143–150, 1999.

[34] D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, pages 2879–2886, 2012.

[35] S. Roberts. Control chart tests based on geometric moving averages. *Technometrics*, 1(3):239–250, 1959.

[36] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *International conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[37] J. M. Saragih, S. Lucey, and J. F. Cohn. Deformable model fitting by regularized landmark mean-shift. *Int. J. Comput. Vision*, 91(2):200–215, 2011.

[38] T. Sugimoto, M. Fukushima, and T. Ibaraki. A parallel relaxation method for quadratic programming problems with interval constraints. *Journal of Computational and Applied Mathematics*, 60(12):219 – 236, 1995.

[39] T. Weise, S. Bouaziz, H. Li, and M. Pauly. Realtime performance-based facial animation. *ACM Trans. Graph.*, 30(4):77:1–77:10, 2011.

[40] T. Weise, H. Li, L. Van Gool, and M. Pauly. Face/off: Live facial puppetry. In *SCA '09*, pages 7–16, 2009.

[41] X. Xiong and F. De la Torre. Supervised descent method and its application to face alignment. In *CVPR*, pages 532–539, 2013.

[42] L. Zhang, N. Snavely, B. Curless, and S. M. Seitz. Spacetime faces: High resolution capture for modeling and animation. *ACM Trans. Graph.*, 23(3):548–558, 2004.