Online Modeling For Realtime Facial Animation

Sofien Bouaziz[*] EPFL

Yangang Wang[†] EPFL / Tsinghua University

Mark Pauly[‡] EPFL

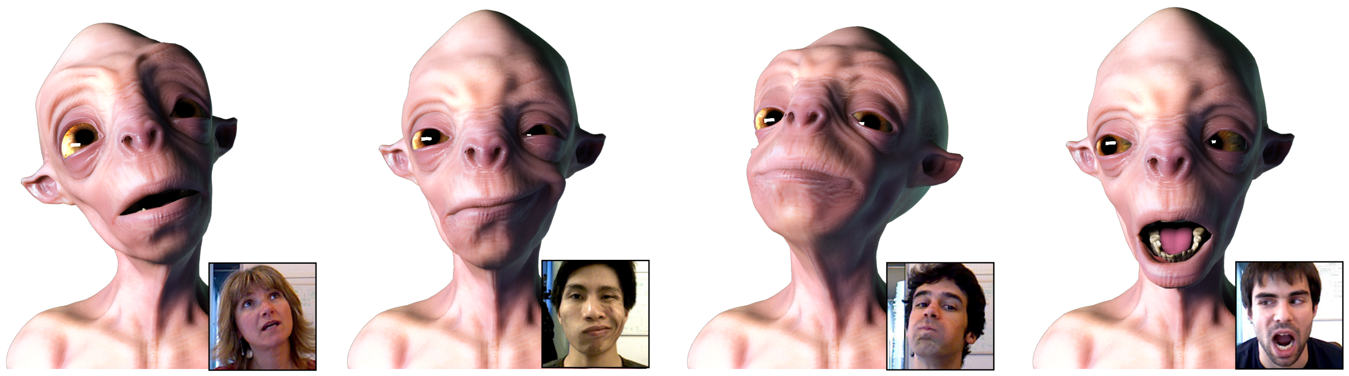


Figure 1: *Realtime tracking and retargeting of the facial expressions of the user (inset) captured with an RGB-D sensor.*

Abstract

We present a new algorithm for realtime face tracking on commodity RGB-D sensing devices. Our method requires no user-specific training or calibration, or any other form of manual assistance, thus enabling a range of new applications in performance-based facial animation and virtual interaction at the consumer level. The key novelty of our approach is an optimization algorithm that jointly solves for a detailed 3D expression model of the user and the corresponding dynamic tracking parameters. Realtime performance and robust computations are facilitated by a novel subspace parameterization of the dynamic facial expression space. We provide a detailed evaluation that shows that our approach significantly simplifies the performance capture workflow, while achieving accurate facial tracking for realtime applications.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: markerless performance capture, face animation, realtime tracking, blendshape animation

Links: DL PDF

[*]sofien.bouaziz@epfl.ch
[†]wang-yg09@mails.tsinghua.edu.cn
[‡]mark.pauly@epfl.ch

ACM Reference Format
Bouaziz, S., Wang, Y., Pauly, M. 2013. Online Modeling For Realtime Facial Animation.
ACM Trans. Graph. 32, 4, Article 40 (July 2013), 9 pages. DOI = 10.1145/2461912.2461976
http://doi.acm.org/10.1145/2461912.2461976.

1 Introduction

Recent advances in realtime performance capture have brought within reach a new form of human communication. Capturing dynamic facial expressions of a user and retargeting these expressions to a digital character in realtime allows enacting arbitrary virtual avatars with live feedback. Compared to communication via recorded video streams that only offer limited ability to alter one's appearance, such technology opens the door to fascinating new applications in computer gaming, social networks, television, training, customer support, or other forms of online interactions.

Successfully deploying such a technology at a large scale puts high demands on performance and usability. Facial tracking needs to be accurate and fast enough to create plausible and responsive animations that faithfully match the performance of the captured user. Ease-of-use affects both hardware and system handling. Marker-based systems, multi-camera capture devices, or intrusive scanners commonly used in high-end animation production are not suitable for consumer-level applications. Equally inappropriate are methods that require complex calibration or necessitate extensive manual assistance to setup or operate the system.

Several realtime methods for face tracking have been proposed that require only a single video camera [Chai et al. 2003; Amberg et al. 2009; Saragih et al. 2011] or consumer-level RGB-D camera, such as the Microsoft Kinect [Weise et al. 2011; Baltrušaitis et al. 2012]. Video-based methods typically track a few facial features and often lack fine-scale detail, which limits the quality of the resulting animations. Tracking performance can also degrade in difficult lighting situations that commonly occur in a home environment, for example. Additionally exploiting 3D depth information obtained by active IR sensing improves tracking accuracy and robustness. This is commonly achieved using a 3D template model [Bradley et al. 2010; Valgaerts et al. 2012] or building a dynamic 3D expression model (DEM) that represents the 3D geometry of the individual facial expressions of the user [Weise et al. 2011]. The DEM allows formulating facial tracking as a non-rigid registration problem in a low-dimensional parameter space, thus facilitating robust and efficient tracking.

However, current methods have one major drawback: The DEM must be created a priori during a controlled training stage, where each user is scanned in several pre-defined expressions. Manual corrections and parameter tuning is often required to achieve satisfactory tracking results. While appropriate for professionals in animation content creation, such user-specific calibration is a severe impediment for deployment in consumer-level applications. We propose an algorithm that addresses this problem.

Contributions. We introduce an adaptive DEM that combines a dynamic expression template, an identity PCA model, and a parameterized deformation model in a low-dimensional representation suitable for online learning. We show how this generic model can be adapted to a specific user on-the-fly without any manual assistance. Our core algorithmic contribution integrates online DEM learning directly into the tracking method. As more and more of the user's expression space is observed during tracking, the generic DEM is progressively adapted to the facial features of the specific user, which in turn will lead to more accurate tracking. Combined with state-of-the-art registration methods, our algorithm yields a fully automatic, realtime face tracking and animation system suitable for consumer-level applications (see Figures 1 and 12).

Related work. Animating digital characters based on facial performance capture is a well-established approach in the computer graphics industry and has been an active area of research. We briefly review the most common methods, but refer to [Pighin and Lewis 2006] for a more detailed discussion. Marker-based systems are widely used to capture realtime performances [Lin and Ouhyoung 2005; Deng et al. 2006]. Explicit face markers significantly simplify tracking, but also limit the amount of spatial detail that can be captured. Performance capture based on dense 3D acquisition, such as structured light scanners [Zhang et al. 2004; Weise et al. 2009] or multi-view camera systems [Furukawa and Ponce 2009; Bradley et al. 2010; Beeler et al. 2011; Valgaerts et al. 2012], have been developed more recently and proven efficient to capture fine-scale dynamics. Processing times can be significant, however, often impeding interactive framerates. Realtime performance can be achieved by a combination of markers and 3D scanning, while still preserving fine-scale spatial and temporal detail [Ma et al. 2008; Bickel et al. 2008; Huang et al. 2011]. However, these systems require specialized hardware setups that need careful calibration.
None of the above methods is suitable or easily adaptable to the kind of consumer-level applications that we target, where minimal hardware setup, realtime performance, and the absence of complex manual calibration or extensive pre-processing are mandatory.
Several realtime systems have been developed that do not require complex hardware or markers, but instead operate with a single camera to record facial performances [Chai et al. 2003; Amberg et al. 2009; Saragih et al. 2011]. The price for this simplification in the acquisition system is a substantially lower tracking quality that often leads to artifacts in the generated face animations. Our goal is to raise tracking quality while keeping the acquisition system simple enough for consumer-level applications and avoiding any manual system calibration or training.
Recent developments in RGB-D technology, such as the Microsoft Kinect or Asus Xtion Live, facilitate this goal. The method presented in [Baltrušaitis et al. 2012] demonstrates how integrating depth and intensity information in a constrained local model improves tracking performance significantly compared to image-based tracking alone. Similarly, the realtime performance-based facial animation system proposed in [Weise et al. 2011] combines 2D and 3D non-rigid registration methods in a single optimization to achieve high-quality tracking. The main drawback of this approach in the context of consumer applications is the need for extensive

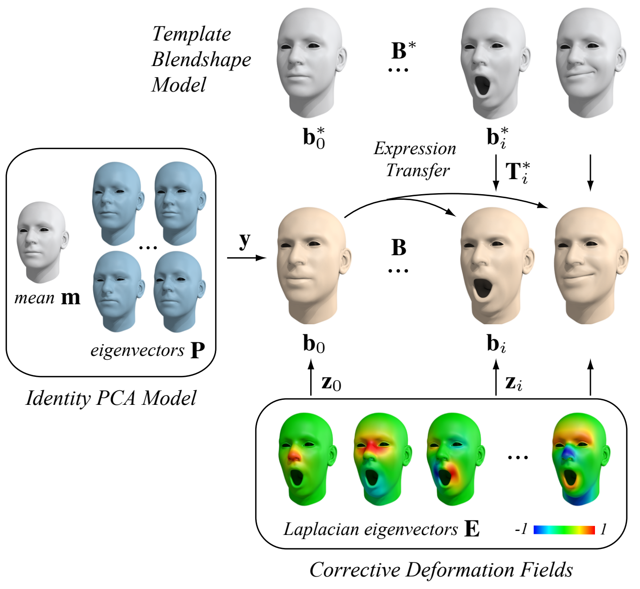


Figure 2: *Adaptive DEM. The user-specific blendshape model* $\boldsymbol{B}$ *is created using a combination of identity PCA model, expression transfer from the template model* $\boldsymbol{B}^*$*, and corrective deformation fields for each blendshape.*

training. Robust and efficient tracking is achieved by building an accurate 3D expression model of the user by scanning and processing a predefined set of facial expressions. Beyond being time-consuming, this preprocess is also error-prone. Users are asked to move their head in front of the sensor in a specific static pose to accumulate sufficient depth information. However, assuming and maintaining the correct pose (e.g. mouth open for a specific, predefined opening angle) is difficult and often requires multiple tries. In contrast, our approach requires no user-specific preprocessing, nor any calibration or user-assisted training, making the tracking system operational right away for any new user.

Overview. The input to our system comes from a consumer-level RGB-D device, such as the Microsoft Kinect or the Asus Xtion Live, that provides a color image and 3D depth map of 640x480 resolution at 30 Hz. Due to the wide-angle lens, the face is confined to a region of about 160x160 pixels. Our goal is to estimate expression parameters that accurately capture the facial dynamics of the observed user in a representation that is appropriate for animating digital avatars. Similar to previous work, e.g. [Weise et al. 2011; Huang et al. 2011], we employ a 3D blendshape model that offers a compact representation suitable for realtime tracking. In our system we build the specific blendshape model of a user concurrently to the tracking optimization, requiring no preceding training or calibration stage. Starting from a rough initial estimate, the dynamic expression model (DEM) is continuously refined as tracking progresses. As soon as each blendshape has been observed sufficiently many times, the DEM converges to a steady state.
We first describe our adaptive DEM that can be customized on the fly to the particular expression space of the user (Section 2). Then in Section 3 we show how realtime tracking can be achieved by registering the DEM with the observed image and depth map data, while concurrently refining the DEM to match the geometry of the observed user. Section 4 provides a detailed evaluation to demonstrate that our realtime performance-based animation system achieves accurate tracking results (see also accompanying video).

2 Adaptive Dynamic Expression Model

Blendshapes. We represent a DEM as a set of blendshape meshes $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_n]$, where $\mathbf{b}_0$ is the neutral pose and the $\mathbf{b}_i, i > 0$ define specific base expressions. All blendshapes have the same static mesh combinatorics and are represented by stacked coordinate vectors. A new facial expression is generated as $\mathbf{F}(\mathbf{x}) = \mathbf{b}_0 + \Delta\mathbf{B}\mathbf{x}$, where $\Delta\mathbf{B} = [\mathbf{b}_1 - \mathbf{b}_0, \ldots, \mathbf{b}_n - \mathbf{b}_0]$, and $\mathbf{x} = [x_1, \ldots, x_n]^T$ are blendshape weights bounded between 0 and 1. The blendshape representation is well suited for realtime performance capture because it reduces tracking to estimating the rigid head alignment and the n blendshape weights for each frame. As an additional benefit, the blendshapes $\mathbf{b}_i$ can be chosen to match pre-defined semantics of common face animation controllers, e.g. *mouth-open*, *smile*, *frown*, etc., which simplifies post-editing and animation retargeting.

We denote with $\mathbf{B}^* = [\mathbf{b}_0^*, \ldots, \mathbf{b}_n^*]$ a template blendshape model that is given a priori, in our case modeled by hand (see additional material for a complete list of blendshapes). This template model defines the expression semantics that we want to transfer onto the DEM of the tracked user during online model building as described in Section 3. Next, we introduce the main ingredients to achieve this dynamic adaptation: an identity PCA model, an expression transfer operator, and corrective deformation fields (Figure 2).

Identity PCA model. We capture variations of face geometry across different users with a morphable model as proposed in [Blanz and Vetter 1999]. Given a large set of meshes of different human faces with one-to-one vertex correspondence in neutral expression, we build a reduced representation using PCA on the stacked vertex coordinate vectors. Let $\mathbf{m}$ be the resulting mean face and $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_l]$ the first l PCA eigenvectors. With such an orthonormal basis, a specific face model in neutral expression can be approximated as $\mathbf{b}_0 = \mathbf{m} + \mathbf{P}\mathbf{y}$ with suitable linear coefficients $\mathbf{y} = [y_1, \ldots, y_l]^T$.

Expression transfer operator. For a given neutral expression $\mathbf{b}_0$ we define approximations for all other blendshapes using a variant of deformation transfer [Sumner and Popović 2004], see also [Li et al. 2010]. Using the template $\mathbf{B}^*$, we transfer the known deformation of the neutral expression $\mathbf{b}_0^*$ to a specific blendshape expression $\mathbf{b}_i^*$, onto the neutral expression $\mathbf{b}_0$ in order to obtain $\mathbf{b}_i$. Our formulation defines $\mathbf{b}_i$ as a linear transformation $\mathbf{T}_i^*\mathbf{b}_0$ of the neutral expression $\mathbf{b}_0$. Contrary to previous formulations of deformation transfer [Sumner and Popović 2004; Botsch et al. 2006], the operator $\mathbf{T}_i^*$ does not depend on $\mathbf{b}_0$, which allows the model refinement optimization to be formulated as the solution to a linear system that can be computed efficiently and robustly (see Section 3.2). A derivation of our expression transfer operator $\mathbf{T}_i^*$ is given in the appendix.

Corrective deformation fields. The PCA model represents the large-scale variability of facial geometries in the neutral expression, but might not capture user-specific details. Similarly, deformation transfer copies expressions from the template without accounting for the particular facial dynamics of the user. We therefore apply additional surface deformation fields to each reconstructed blendshape mesh $\mathbf{b}_i \in \mathbf{B}$ to obtain a more faithful reconstruction of the user's facial expression space.
Per-vertex displacements are modeled using a spectral representation defined by the k last eigenvectors $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_k]$ of the graph Laplacian matrix $\mathbf{L}$ computed on the 3D face mesh, see [Levy and Zhang 2010] for more details. A smooth deformation field can then be defined as a linear combination $\mathbf{E}\mathbf{z}$, where $\mathbf{z} = [z_1, \ldots, z_k]^T$ are the spectral coefficients. The spectral basis offers two main advantages in our setting: We can optimize for the corrective deformations in a low-dimensional space, requiring only k variables to represent a deformation of a blendshape mesh. In addition, the built-in smoothness of the low-frequency eigenvectors helps to avoid overfitting when aligning the blendshapes to noisy depth maps.

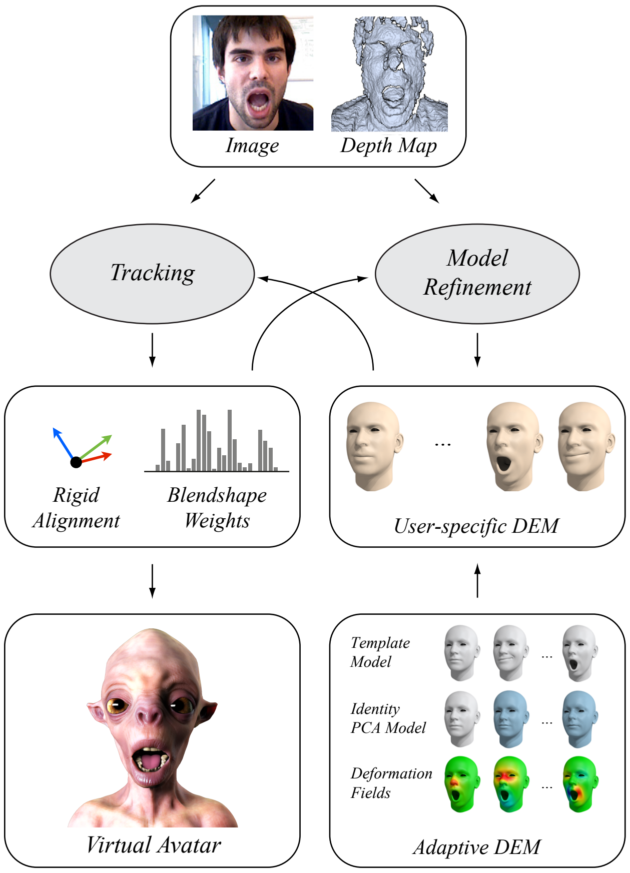


Figure 3: *Optimization pipeline. Each frame of the input data (color image and depth map), is processed with our interleaved optimization that alternates tracking and model refinement. The output are tracking parameters (rigid alignment, blendshape weights) per frame that can be used to drive a virtual avatar in realtime. Concurrently, the user-specific DEM is adapted according to the facial characteristics of the observed user.*

Parameterized DEM. With all this machinery in place, we can now define a parameterized DEM that can be adapted to a particular user (see Figure 2). The neutral expression is given as $\mathbf{b}_0 = \mathbf{m} + \mathbf{P}\mathbf{y} + \mathbf{E}\mathbf{z}_0$, i.e., a combination of identity PCA model and a corrective deformation field. The remaining blendshapes $\mathbf{b}_1, \ldots, \mathbf{b}_n$ are parameterized as

$$\mathbf{b}_i = \mathbf{T}_i^*\mathbf{b}_0 + \mathbf{E}\mathbf{z}_i = \mathbf{T}_i^*(\mathbf{m} + \mathbf{P}\mathbf{y} + \mathbf{E}\mathbf{z}_0) + \mathbf{E}\mathbf{z}_i,$$

i.e., combining expression transfer of the template $\mathbf{B}^*$ to the neutral expression $\mathbf{b}_0$ with expression-specific deformation fields.

3 Optimization

The adaptive DEM described in the previous section is at the core of our tracking optimization algorithm. The goal of this optimization is to compute accurate tracking parameters, while at the same time refining the user-specific DEM in realtime.

More precisely, our algorithm solves for

- the rigid alignment of the face model to the input depth map defined by a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ at each frame t,
- the blendshape weights $\mathbf{x} = [x_1, \dots, x_n]^T$ for each frame t,
- the identity PCA parameters $\mathbf{y} = [y_1, \dots, y_l]^T$ for the neutral face expression $\mathbf{b}_0$ of the user, and
- the deformation coefficients $\mathbf{Z} = \{\mathbf{z}_0, \dots, \mathbf{z}_n\}$ for each blendshape $\mathbf{b}_i$, where $\mathbf{z}_i = [z_{i,1}, \dots z_{i,k}]^T$.

We use superscripts to refer to specific time frames, e.g. $\mathbf{x}^t$ denotes the blendshape weights at frame $t \in \mathbb{N}$, where $t = 1$ denotes the first frame. To simplify notation, we omit the superscripts when irrelevant or clear from the context.

The optimization alternates between two stages as shown in Figure 3. Stage I estimates the rigid alignment and blendshape weights, keeping the DEM fixed. Stage II refines the user-specific DEM by solving for the PCA parameters $\mathbf{y}$ and deformation coefficients $\mathbf{Z}$, keeping the blendshape weights fixed. We bootstrap this alternating minimization by initializing the DEM with the PCA reconstruction for the neutral expression and deformation transfer of the template DEM as described next.

Initialization. Our system requires the user to enter the sensor's field of view in a neutral facial expression. We use the method of [Viola and Jones 2001] to detect the face and crop the depth map to obtain a 3D scan of the neutral expression. From this initial face scan, we compute a first approximation of $\mathbf{b}_0$ by aligning the parameterized neutral expression to the depth map. This means that we solve for the PCA coefficients $\mathbf{y}$ and deformation coefficients $\mathbf{z}_0$, as well as the rigid head pose $(\mathbf{R}, \mathbf{t})$, by minimizing the common ICP energy with point-plane constraints [Rusinkiewicz and Levoy 2001]. More specifically, we solve for

$$\arg\min_{\mathbf{R},\mathbf{t},\mathbf{y},\mathbf{z}_0} \|\mathbf{A}_0(\mathbf{R}\mathbf{b}_0+\mathbf{t})-\mathbf{c}_0\|_2^2+\beta_1\|\mathbf{D_P}\mathbf{y}\|_2^2+\beta_2\|\mathbf{D_E}\mathbf{z}_0\|_2^2+\beta_3\|\mathbf{z}_0\|_2^2. \quad (1)$$

Here $(\mathbf{A}_0, \mathbf{c}_0)$, is the matrix resp. right-hand side summarizing the ICP constraint equations in the first term of the objective function (see [Weise et al. 2011] for details). The remaining summands are regularization terms with corresponding positive scalar weights β_1, β_2, β_3. The term $\mathbf{D_P}\mathbf{y}$ regularizes the PCA weights, where $\mathbf{D_P}$ is a diagonal matrix containing the inverse of the standard deviation of the PCA basis. The term $\mathbf{D_E}\mathbf{z}_0$ regularizes the deformation coefficients by measuring the bending of the deformation. $\mathbf{D_E}$ is the diagonal matrix of eigenvalues corresponding to the eigenvectors in $\mathbf{E}$ of the Laplacian matrix $\mathbf{L}$ [Botsch et al. 2010]. The last summand penalizes the magnitude of the deformation vectors.

The optimization is solved using the Gauss-Newton method [Madsen et al. 2004]. We initialize the solver with $\mathbf{y} = \mathbf{z}_0 = 0$, the initial face location is retrieved from the face detector with the user assumed to be front-facing. Given the reconstruction of $\mathbf{b}_0^1$ at the first frame ($t = 1$), we initialize the additional blendshapes by applying the deformation transfer operator, i.e. $\mathbf{b}_i^1 = \mathbf{T}_i^* \mathbf{b}_0^1$ for $i = 1, \dots, n$.

3.1 Tracking

The tracking stage of the optimization assumes that the DEM is fixed and solves for the rigid motion $(\mathbf{R}, \mathbf{t})$ and blendshape weights $\mathbf{x}$ at timeframe t.

Rigid motion tracking. We first estimate $\mathbf{R}$ and $\mathbf{t}$ by directly aligning the static reconstructed mesh of the previous frame with the acquired depth map of the current frame using ICP with point-plane constraints. To stabilize the rigid motion, the constraints are only defined for the front head and nose region of the reconstructed mesh as illustrated in blue on the right.

Estimating blendshape weights. Given the rigid pose and the current set of blendshapes $\mathbf{B}$, we now need to estimate the blendshape weights $\mathbf{x}$ that best match the input data of the current frame. We formulate this problem as a combined 2D/3D registration. The 2D registration is formulated using optical flow constraints, while the 3D registration is using ICP as above. This yields a fitting energy of the form

$$E_{\text{fit}} = \|\mathbf{A}(\mathbf{b}_0 + \Delta\mathbf{B}\mathbf{x}) - \mathbf{c}\|_2^2, \quad (2)$$

where $(\mathbf{A}, \mathbf{c})$ summarize the registration constraints on a subset of the face vertices as indicated in blue on the left. For brevity we omit the specific formulas here, detailed derivations of the constraint terms can be found in [Weise et al. 2011]. Our optimization iteratively minimizes the following energy

$$\arg\min_{\mathbf{x}} E_{\text{fit}} + \lambda_1 E_{\text{smooth}} + \lambda_2 E_{\text{sparse}}. \quad (3)$$

Two additional terms, E_{smooth} and E_{sparse} with non-negative weights λ_1 and λ_2, are added for regularization. Temporal smoothness is enforced by penalizing the second-order difference

$$E_{\text{smooth}} = \|\mathbf{x}^{t-2} - 2\mathbf{x}^{t-1} + \mathbf{x}^t\|_2^2,$$

where t denotes the current timeframe. We also apply the 1-norm regularization

$$E_{\text{sparse}} = \|\mathbf{x}\|_1$$

on the blendshape coefficients. We found that this sparsity-inducing energy is very important to stabilize the tracking (see Figure 4). Because the blendshape basis are not linearly independent, the same expression could in principle be represented by different blendshape combinations. Favoring a reconstruction with as few blendshapes as possible avoids potential blendshape compensation artifacts and better matches the blendshape weights a human animator would chose, which can be advantageous for retargeting. In addition, the l_1 regularization leads to a significant speed-up of the subsequent model refinement stage (Section 3.2), since blendshape refinement is only performed on blendshapes with non-zero blendshape weight (see also Figure 6).
The optimization is performed using a warm started shooting method [Fu 1998]. The blendshape weights $\mathbf{x} = [x_1, \dots, x_n]^T$ are bounded between 0 and 1 by projection over the constraint set at each iteration.

3.2 DEM Refinement

The refinement stage of the optimization adapts the blendshape model by solving for the PCA parameters $\mathbf{y}$ and deformation coefficients $\mathbf{z}_0, \dots, \mathbf{z}_n$, keeping the rigid pose $(\mathbf{R}, \mathbf{t})$ and the blendshape weights $\mathbf{x}$ computed in the previous stage fixed.

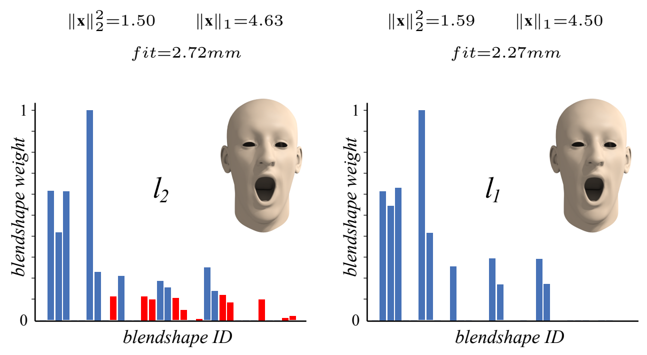


Figure 4: *Comparison between l_1 and l_2 regularization for the blendshape weight optimization of Equation 3. The l_1 regularization leads to a lower average fitting error (denoted by fit), but more importantly, significantly reduces the number of non-zero blendshape weights. The red bars on the left show the additionally activated blendshapes under l_2 norm regularization.*

We rewrite the fitting energy in Equation 2 as

$$E_{\text{fit}} = \|\mathbf{A}(\mathbf{b}_0 + \Delta\mathbf{B}\mathbf{x}) - \mathbf{c}\|_2^2 = \|\mathbf{A}[\bar{x}\mathbf{b}_0 + \sum_{i=1}^{n} x_i \mathbf{b}_i] - \mathbf{c}\|_2^2,$$

where $\bar{x} = 1 - \sum_{i=1}^{n} x_i$. With $\mathbf{b}_0 = \mathbf{m} + \mathbf{P}\mathbf{y} + \mathbf{E}\mathbf{z}_0$ and $\mathbf{b}_i = \mathbf{T}_i^* \mathbf{b}_0 + \mathbf{E}\mathbf{z}_i$, this term can then be reformulated as $E_{\text{fit}} = \|\bar{\mathbf{A}}\mathbf{u} - \bar{\mathbf{c}}\|_2^2$, where

$$\bar{\mathbf{A}} = \mathbf{A}[(\bar{x}\mathbf{I} + \sum_{i=1}^{n} x_i \mathbf{T}_i^*)\mathbf{P}, (\bar{x}\mathbf{I} + \sum_{i=1}^{n} x_i \mathbf{T}_i^*)\mathbf{E}, x_1\mathbf{E}, \ldots, x_n\mathbf{E}],$$

$$\mathbf{u} = [\mathbf{y}^T, \mathbf{z}_0^T, \ldots, \mathbf{z}_n^T]^T, \quad \text{and} \quad \bar{\mathbf{c}} = \mathbf{c} - \mathbf{A}(\bar{x}\mathbf{I} + \sum_{i=1}^{n} x_i \mathbf{T}_i^*)\mathbf{m}.$$

As previously, we regularize the PCA coefficients $\mathbf{y}$ and deformation coefficients $\mathbf{z}_i$, leading to the model refinement energy

$$E_{\text{ref}} = \|\bar{\mathbf{A}}\mathbf{u} - \bar{\mathbf{c}}\|_2^2 + \beta_1\|\mathbf{D}_{\mathbf{P}}\mathbf{y}\|_2^2 + \sum_{i=0}^{n}(\beta_2\|\mathbf{D}_{\mathbf{E}}\mathbf{z}_i\|_2^2 + \beta_3\|\mathbf{z}_i\|_2^2). \quad (4)$$

Temporal Aggregation. The optimization of the DEM should not only depend on the current frame, but consider the entire history of observed expressions. However, directly optimizing over all frames would quickly become prohibitive in terms of memory and computation overhead. We therefore introduce an aggregation scheme that keeps the memory cost constant. The optimization is formulated as

$$\underset{\mathbf{y},\mathbf{z}_0,\ldots,\mathbf{z}_n}{\arg\min} \sum_{j=1}^{t} \frac{\gamma^{t-j}}{\sum_{j=1}^{t}\gamma^{t-j}} E_{\text{ref}}^j, \quad (5)$$

where t is the current frame and $0 \leq \gamma \leq 1$ defines an exponential decay over the frame history. E_{ref}^j denotes the refinement energy of Equation 4 at time j. The optimal solution of this minimization can be found by solving

$$(\mathbf{D} + \sum_{j=1}^{t} \frac{\gamma^{t-j}}{\sum_{j=1}^{t}\gamma^{t-j}} (\bar{\mathbf{A}}^j)^T \bar{\mathbf{A}}^j)\mathbf{u} = \sum_{j=1}^{t} \frac{\gamma^{t-j}}{\sum_{j=1}^{t}\gamma^{t-j}} (\bar{\mathbf{A}}^j)^T \bar{\mathbf{c}}^j, \quad (6)$$

where $\mathbf{D}$ is a diagonal matrix containing the regularization terms of Equation 4. To solve this system, we propose an online algorithm based on warm-started Gauss-Seidel optimization [Barrett et al. 1994]. Our algorithm allows optimizing over the entire history of frames with a fixed memory overhead, as we do not need to store each frame separately (see Algorithm 1).

Algorithm 1: Blendshape Refinement at frame t

```
1 Initialization: $\mathbf{M}^1 = \mathbf{0}$, $\mathbf{y}^1 = \mathbf{0}$, $s^1 = 0$
2 $s^t = \gamma s^{t-1} + 1$
3 $\mathbf{M}^t = \gamma\frac{s^{t-1}}{s^t}\mathbf{M}^{t-1} + \frac{1}{s^t}(\bar{\mathbf{A}}^t)^T\bar{\mathbf{A}}^t$
4 $\mathbf{y}^t = \gamma\frac{s^{t-1}}{s^t}\mathbf{y}^{t-1} + \frac{1}{s^t}(\bar{\mathbf{A}}^t)^T\bar{\mathbf{c}}^t$
5 Output: $\mathbf{u}^t = GaussSeidel(\mathbf{M}^t + \mathbf{D}, \mathbf{y}^t, \mathbf{u}^{t-1})$
```

Figure 5 illustrates the tradeoff between fitting error and temporal variance as a function of the parameter γ. We found $\gamma = 0.9$ to provide a good balance and use this value for all our experiments.

Blendshape coverage. In principle, DEM refinement could run indefinitely to continuously optimize the blendshape model as tracking progresses. However, we can improve computational performance with a simple heuristic. Blendshapes that have been optimized sufficiently many times can be considered "saturated" and are removed from the optimization. We define a coverage coefficient $\sigma_i = \sum_{j=1}^{t} x_i^j$ that measures how well each blendshape $\mathbf{b}_i$

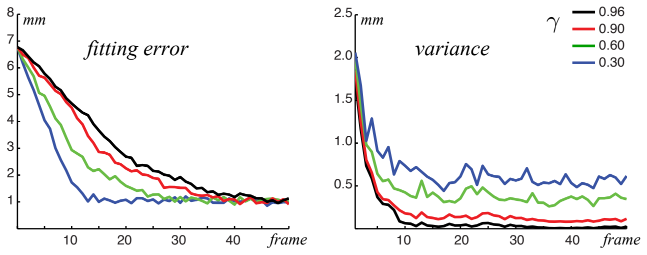


Figure 5: *Effect of the temporal decay factor γ in Equation 5. Lower values lead to faster reduction in fitting error, measured as the mean non-rigid ICP error for each frame, but incur more variance, measured as the mean per-vertex difference between consecutive frames.*

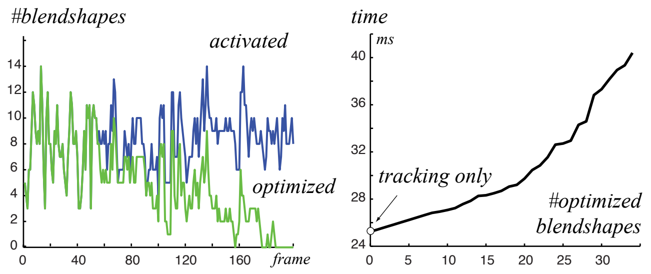


Figure 6: *Optimization performance. Left: The number of blendshapes optimized during DEM refinement gradually decreases as more blendshapes reach the coverage threshold. Right: total computation time per frame as a function of the number of blendshapes that are optimized in each frame.*

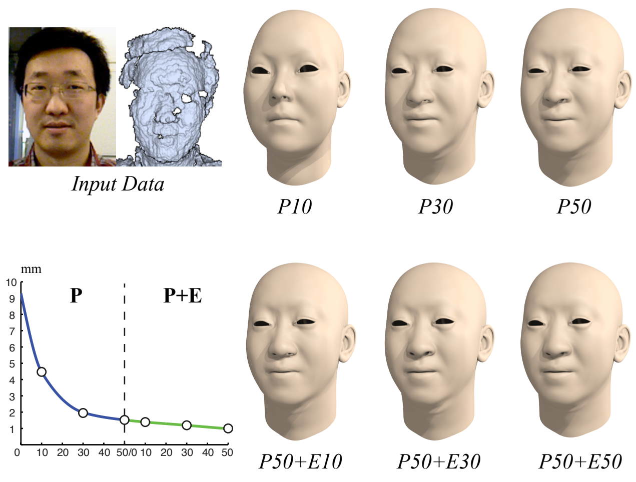


Figure 7: *Evaluation of the initial estimation of the neutral expression $\mathbf{b}_0$ when varying the number of PCA basis in $\mathbf{P}$ and the number of Laplacian eigenvector in $\mathbf{E}$. The graph shows the mean non-rigid ICP error averaged over a sequence of 440 frames.*

has been observed until the current frame t. As soon as $\sigma_i > \bar{\sigma}$ for some fixed threshold $\bar{\sigma}$, the corresponding blendshape $\mathbf{b}_i$ is considered saturated and remains constant for the subsequent optimization. Since the neutral expression $\mathbf{b}_0$ plays a special role as the source for expression transfer, we always run the full optimization until $\sum_{j=1}^{t} \max(\bar{x}^j, 0) > \bar{\sigma}$. In practice, this does not affect performance significantly, since $\mathbf{b}_0$ is the blendshape that is typically most often observed. Figure 6 gives an indication of the computational overhead of DEM refinement. Since the computational cost gradually decreases as more blendshapes reach their coverage thresholds, DEM refinement quickly becomes negligible compared to the tracking stage of the optimization.

3.3 Implementation

In our current implementation, we employ a blendshape model of 34 blendshapes (see additional material for a complete list). The identity PCA model is computed from the dataset of [Blanz and Vetter 1999] that consists of 100 male and 100 female head scans of young adults. We use 50 PCA basis vectors to approximate the neutral expression for all our examples. The corrective deformation fields are represented by 50 Laplacian eigenvectors for each coordinate (see also Section 4). We empirically determined the parameters $\beta_1 = 0.5$, $\beta_2 = 0.1$, and $\beta_3 = 0.001$ for Equations 1 and 4, $\lambda_1 = 10$ and $\lambda_2 = 20$ for Equation 3, and $\bar{\sigma} = 10$ for the coverage threshold, and use the same fixed settings for all our examples.

Our software is implemented in C++ and parallelized using OpenMP. We use the Eigen library for linear algebra computations and OpenCV for the face detector [Viola and Jones 2001] and image processing operations. In order to speed-up the system we do not optimize for the unknowns of the blendshapes with 0 weight, keeping them fixed during the Gauss-Seidel optimization. Another speed improvement is achieved by building $(\bar{\mathbf{A}}^t)^T\bar{\mathbf{A}}^t$ per block as numerous blocks are similar up to a scalar factor, and blocks corresponding to the blendshapes with 0 weights are 0.

To complete the face tracking algorithm, we have implemented a separate image-based eye tracker. Since the rigid and the non-rigid alignment accurately determine the location of the eyes in the color image, we can apply a k-nearest neighbor search in a database of labeled eyes by cropping, rectifying and normalizing the input image. This k-nearest neighbor search is implemented using the OpenCV library. The final result is a weighted average of the labels of the k neighbors. The result of the eye tracker drives 14 supplementary blendshapes (see additional material) localized around the eyes. These blendshapes are computed using expression transfer only and are not part of the model refinement optimization.
Our system achieves sustained framerates of 25 Hz with a latency of 150 ms on a MacBook Pro with an Intel Core i7 2.7Ghz processor, 16 GBytes of main memory, and an NVIDIA GeForce GT 650M 1024MB graphics card.

4 Evaluation

In this section we present several experiments that we have performed to analyze our optimization algorithm, and discuss limitations of our approach.

Dynamic expression model. Figure 7 shows how the optimization of the neutral face depends on the number of basis vectors used for the identity PCA model and the corrective deformation fields, respectively. Due to the limited number of input samples (200 head models total), we observed no significant improvement beyond 50 basis vectors for the PCA model. For the deformation fields we found that 50 Laplacian eigenvectors are sufficient to obtain accurate reconstructions while still enabling realtime performance. The effect of the deformation fields is also shown for several blendshapes in Figure 8, where notable changes can be observed in the mouth region and around the nostrils. In general, we found these per-vertex deformations to be important to capture geometric detail, in particular the asymmetries common in many faces.

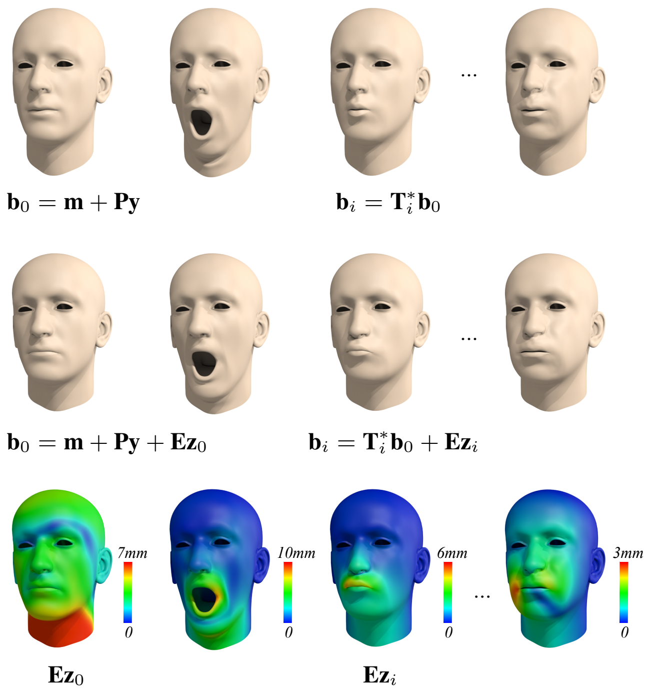

$\mathbf{b}_0 = \mathbf{m} + \mathbf{P}\mathbf{y}$
$\mathbf{b}_i = \mathbf{T}_i^*\mathbf{b}_0$
$\mathbf{b}_0 = \mathbf{m} + \mathbf{P}\mathbf{y} + \mathbf{E}\mathbf{z}_0$
$\mathbf{b}_i = \mathbf{T}_i^*\mathbf{b}_0 + \mathbf{E}\mathbf{z}_i$
7mm
10mm
6mm
3mm
0
$\mathbf{E}\mathbf{z}_0$
$\mathbf{E}\mathbf{z}_i$

Figure 8: *Effect of corrective deformation fields. PCA and expression transfer only (top), additional deformation fields for both $\mathbf{b}_0$ and the $\mathbf{b}_i$ (middle), color-coded vertex displacements due to the deformation fields $\mathbf{E}\mathbf{z}_i$ (bottom).*

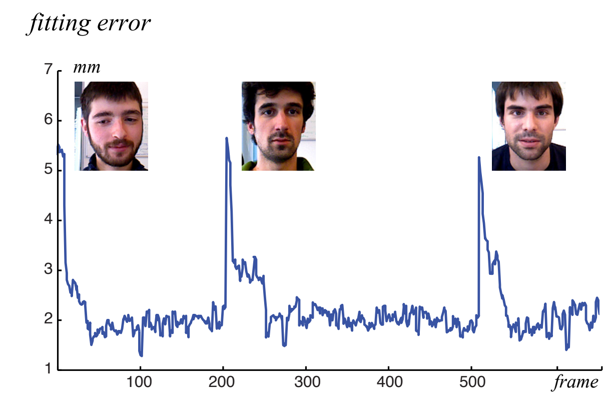


Figure 9: *Dynamic adaptation of the DEM model for three different users. The vertical spikes in fitting error indicate when a new user enters the field of view of the sensor. The DEM quickly adapts to the new facial geometry. High tracking accuracy is typically achieved within a second of using the system.*

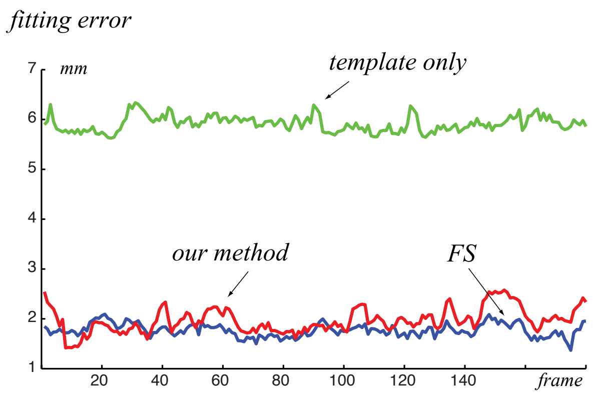


Figure 11: *Comparison of average fitting error for different tracking methods. DEM refinement significantly improves tracking accuracy compared to tracking with the template only. After convergence of the DEM, our method is comparable to the commercial software Faceshift Studio (FS) that depends on user-specific training. For this test, FS requires 11 static face scans of the user to create the expression model, as well as some manual work to assist the reconstruction, while our approach is completely automatic.*

Tracking and DEM refinement. The accompanying video best demonstrates the quality of tracking achieved by our system. Figure 5 and Figure 9 show how the fitting error decreases over time as the DEM is refined concurrently to tracking. The corresponding adaptation of the blendshapes is illustrated in Figure 10. Figure 11 provides a comparison with a commercial software [Faceshift 2013] that requires significant user-specific training and manual assistance to create the DEM. As the plot illustrates, our approach achieves comparable accuracy, while requiring no training or pre-processing.

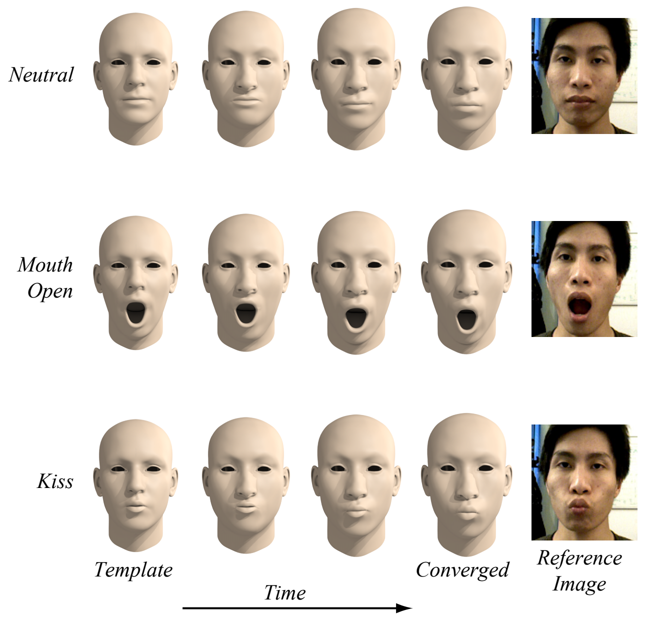


Figure 10: *Progressive DEM refinement. Each row shows the temporal evolution of a specific blendshape. The input image on the right is provided for reference. For this experiment we omit the PCA initialization to illustrate the robustness of the DEM refinement even when large deformations are required to match the face geometry of the tracked user.*

Retargeting. The expression transfer operator (see Section 2 and Appendix), ensures that the user-specific DEM retains the blendshape semantics of the template model. The blendshape weights computed during tracking can therefore be used directly to drive a compatible face rig with the same blendshape configuration. This simple retargeting incurs no extra cost, which is particularly important for realtime applications. Figures 1 and 12 show that virtual avatars with significantly different facial features than the tracked user can be animated faithfully with our method.

Limitations. Our performance capture system is limited by the resolution and noise levels of the input device. While future hardware developments are likely to improve the performance of our system, tracking accuracy is inherently limited by the geometric detail of the template blendshape model. This representation is built on the premise that the location of facial features is spatially consistent across different users, an assumption that is no longer valid at small scales. For example, wrinkles typically appear at different locations for different people. As a consequence, such fine-scale features are not modeled adequately with our current approach.

In general, the same dynamic expression template might not be optimal for all tracked users. For example, children have significantly different facial dynamics than adults, and it might be more appropriate to apply different template models to different age groups. Since the identity PCA model of [Blanz and Vetter 1999] that we currently use does not contain any children or older people, we did not yet investigate this hypothesis further.

The template blendshape model we currently use has been created in an iterative, empirical process. Starting with the most commonly used expressions, such as *mouth open*, *smile*, etc., we successively extended the model to include more blendshapes to obtain a more accurate expression space. This extension has been done with the advice of professional animators in order to match the established conventions for blendshape controllers. However, what constitutes the optimal blendshape model for tracking is not clear. A systematic study answering this question could be interesting future work.

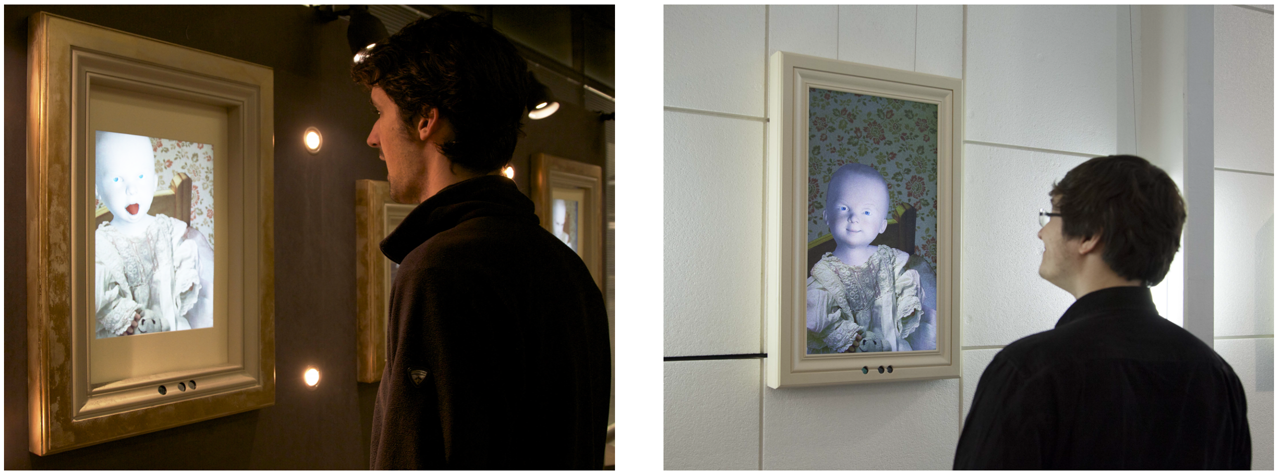


Figure 12: Mimicry*, an application case study using our approach. An observer can simply step in front of the picture frame and the character depicted in the virtual painting will start mimicking the person's facial expression in realtime. The sensor is embedded in the frame. See also accompanying video.*

5 Conclusion

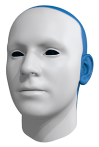
We have demonstrated that online model building can replace user-specific training and manual calibration for facial performance capture systems, while maintaining high tracking accuracy. Requiring only a low-cost 3D sensor and no manual assistance of any kind, our system opens the door for new applications in human communication, such as in-game puppetry, virtual avatars for social networks, or computer-assisted realtime training applications.
Beyond exploring these applications, future work will focus on further improving tracking accuracy. One interesting possibility would be to integrate speech analysis for better lip synching. Another promising avenue for future work is online avatar creation. The user-specific DEM that we build automatically already constitutes a fully rigged geometric avatar. Adding reconstruction of texture and other facial features such as hair would allow building complete digital avatars that can directly be integrated into online applications. Finally, we would like to investigate the application of similar online optimizations to other linear models such as Active Appearance Models or Active Shape Models.

Aknowlegments. We thank Mario Botsch, Bailin Deng, and Boris Neubert for their valuable comments, Brian Amberg, Thibaut Weise and faceshift AG [Faceshift 2013] for their help and support. We are grateful to Minh Dang, Eva Darulova, Mario Deuss, Ivan Dokmanic, Stefan Lienhard, Giuliano Losa, Madeleine Robert, Yuliy Schwartzburg, Feng Yang, and Wei Li for being great actors and to all the other people who took part in the experiments. We are thankful to Emilie Tappolet, Raphaël Muñoz and the HEAD of Geneva for the design of the Mimicry installation. This research is supported by the Tsinghua visiting doctoral students foundation 2012 and the Swiss National Science Foundation grant 20PA21L_129607.

Appendix: Expression Transfer

In this section we describe our formulation of deformation transfer that we use to deform the neutral expression $\mathbf{b}_0$ to an expression $\mathbf{b}_i$ by transferring the deformation from the neutral expression $\mathbf{b}_0^*$ to the expression $\mathbf{b}_i^*$ of a template model. We first compute the set of affine transformations $\{\mathbf{S}_1^*, \ldots, \mathbf{S}_p^*\}$ deforming the p triangles of $\mathbf{b}_0^*$ to the corresponding ones of $\mathbf{b}_i^*$. As an affine transformation is not fully characterized by the deformation of a triangle we instead use tetrahedrons to compute the affine transformations where the fourth vertex is added in the direction perpendicular to the triangle [Sumner and Popović 2004]. The affine transformation $\mathbf{S}^*$ of a tetrahedron $\{\mathbf{v}_{01}^*, \mathbf{v}_{02}^*, \mathbf{v}_{03}^*, \mathbf{v}_{04}^*\}$ of $\mathbf{b}_0^*$ to the corresponding tetrahedron $\{\mathbf{v}_{i1}^*, \mathbf{v}_{i2}^*, \mathbf{v}_{i3}^*, \mathbf{v}_{i4}^*\}$ of $\mathbf{b}_i^*$ is computed as $\mathbf{S}^* = \mathbf{S}_i^* \mathbf{S}_0^{*(-1)}$, where $\mathbf{S}_i^* = [\mathbf{v}_{i2}^* - \mathbf{v}_{i1}^*, \mathbf{v}_{i3}^* - \mathbf{v}_{i1}^*, \mathbf{v}_{i4}^* - \mathbf{v}_{i1}^*]$ and $\mathbf{S}_0^* = [\mathbf{v}_{02}^* - \mathbf{v}_{01}^*, \mathbf{v}_{03}^* - \mathbf{v}_{01}^*, \mathbf{v}_{04}^* - \mathbf{v}_{01}^*]$. The deformation transfer problem can then be formulated as

$$\arg\min_{\mathbf{b}_i} \sum_{j=1}^{p} \|\mathbf{S}_j^* \mathbf{t}_{0j} - \mathbf{t}_{ij}\|_2^2 + \mu \|\mathbf{F}(\mathbf{b}_i - \mathbf{b}_0)\|_2^2,$$



where $\mathbf{t}_{ij} = [\mathbf{v}_{i2} - \mathbf{v}_{i1}, \mathbf{v}_{i3} - \mathbf{v}_{i1}]_j$ represents two edges of the triangle j of $\mathbf{b}_i$, $\mathbf{F}$ is a diagonal matrix defining the vertices that need to be fixed between $\mathbf{b}_0$ and $\mathbf{b}_i$ shown in blue on the left, and μ is a weight factor that we fix to $\mu = 100$ for all our computations. This optimization can be reformulated as

$$\arg\min_{\mathbf{b}_i} \|\mathbf{H}_i^* \mathbf{G} \mathbf{b}_0 - \mathbf{G} \mathbf{b}_i\|_2^2 + \mu \|\mathbf{F}(\mathbf{b}_i - \mathbf{b}_0)\|_2^2, \qquad (7)$$

where $\mathbf{G}$ is a matrix transforming vertices to edges and $\mathbf{H}_i^*$ is a matrix containing the affine transformations mapping each edge of the template neutral expression $\mathbf{b}_0^*$ to the template expression $\mathbf{b}_i^*$. The optimal solution of this problem is $\mathbf{b}_i = \mathbf{T}_i^* \mathbf{b}_0$ where $\mathbf{T}_i^* = (\mathbf{G}^T \mathbf{G} + \mathbf{F})^{-1} (\mathbf{G}^T \mathbf{H}_i^* \mathbf{G} + \mathbf{F})$ is a linear operator defining the transformation from the neutral expression $\mathbf{b}_0$ to an expression $\mathbf{b}_i$ that matches the transformation of $\mathbf{b}_0^*$ to $\mathbf{b}_i^*$. The main difference of our formulation compared to previous approaches proposed in [Sumner and Popović 2004] or [Botsch et al. 2006] is that $\mathbf{T}_i^*$ does not depend on $\mathbf{b}_0$. Effectively, our formulation uses a graph Laplacian instead of a cotan Laplacian, which avoids the weighting factor of triangle areas of $\mathbf{b}_0$ in $\mathbf{T}_i^*$, which would make $\mathbf{T}_i^* \mathbf{b}_0$ non-linear with respect to $\mathbf{b}_0$. Since our face meshes are uniformly tessellated, this simplification has little effect on the resulting deformations (see Figure 13). However, it allows the DEM refinement optimization to be formulated as a simple linear system (see Equation 4) which makes it fast and robust to solve.

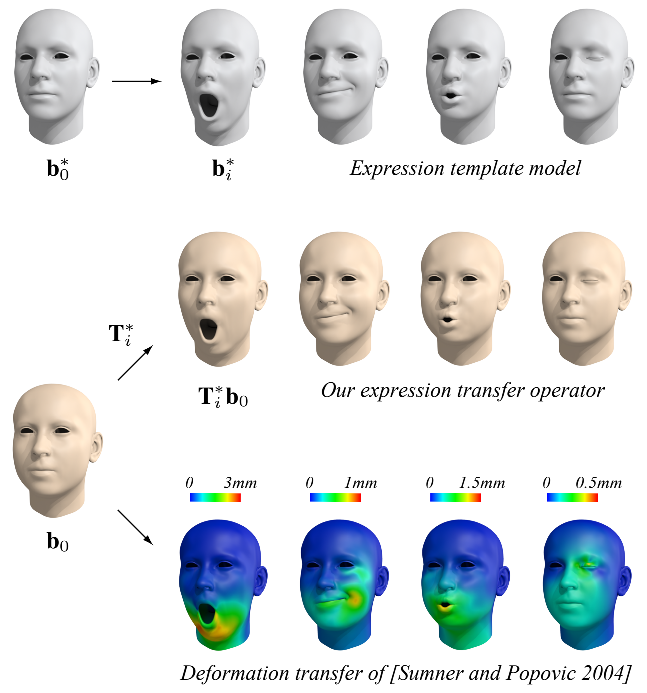

$\mathbf{b}_0^*$
$\mathbf{b}_i^*$
Expression template model
$\mathbf{T}_i^*$
$\mathbf{T}_i^*\mathbf{b}_0$
Our expression transfer operator
$\mathbf{b}_0$
0 3mm
0 1mm
0 1.5mm
0 0.5mm
Deformation transfer of [Sumner and Popovic 2004]

Figure 13: *Expression transfer from a template model (top) to the user-specific model (middle). Our approach gives comparable results to the method of [Sumner and Popovic 2004] (bottom), but can express the transfer operation as a linear transformation.*

References

AMBERG, B., BLAKE, A., AND VETTER, T. 2009. On compositional image alignment, with an application to active appearance models. In *CVPR*, 1714–1721.

BALTRUŠAITIS, T., ROBINSON, P., AND MORENCY, L.-P. 2012. 3d constrained local model for rigid and non-rigid facial tracking. In *CVPR*, 2610–2617.

BARRETT, R., BERRY, M., CHAN, T. F., DEMMEL, J., DONATO, J., DONGARRA, J., EIJKHOUT, V., POZO, R., ROMINE, C., AND DER VORST, H. V. 1994. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM.

BEELER, T., HAHN, F., BRADLEY, D., BICKEL, B., BEARDSLEY, P., GOTSMAN, C., SUMNER, R. W., AND GROSS, M. 2011. High-quality passive facial performance capture using anchor frames. *ACM Trans. Graph. 30*, 75:1–75:10.

BICKEL, B., LANG, M., BOTSCH, M., OTADUY, M. A., AND GROSS, M. 2008. Pose-space animation and transfer of facial details. In *SCA*, 57–66.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3d faces. In *Proc. SIGGRAPH*, 187–194.

BOTSCH, M., SUMNER, R., PAULY, M., AND GROSS, M. 2006. Deformation Transfer for Detail-Preserving Surface Editing. In *VMV*, 357–364.

BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., AND LEVY, B. 2010. *Polygon Mesh Processing*. AK Peters.

BRADLEY, D., HEIDRICH, W., POPA, T., AND SHEFFER, A. 2010. High resolution passive facial performance capture. *ACM Trans. Graph. 29*, 41:1–41:10.

CHAI, J., XIAO, J., AND HODGINS, J. 2003. Vision-based control of 3d facial animation. In *Proc. SCA*, 193–206.

DENG, Z., CHIANG, P.-Y., FOX, P., AND NEUMANN, U. 2006. Animating blendshape faces by cross-mapping motion capture data. In *I3D*, 43–48.

FACESHIFT, 2013. `http://www.faceshift.com`.

FU, W. J. 1998. Penalized Regressions: The Bridge versus the Lasso. *J. Comp. Graph. Stat. 7*, 397–416.

FURUKAWA, Y., AND PONCE, J. 2009. Dense 3d motion capture for human faces. In *CVPR*, 1674–1681.

HUANG, H., CHAI, J., TONG, X., AND WU, H. 2011. Leveraging motion capture and 3d scanning for high-fidelity facial performance acquisition. *ACM Trans. Graph. 30*, 74:1–74:10.

LEVY, B., AND ZHANG, R. H. 2010. Spectral geometry processing. In *ACM SIGGRAPH Course Notes*.

LI, H., WEISE, T., AND PAULY, M. 2010. Example-based facial rigging. *ACM Trans. Graph. 29*, 32:1–32:6.

LIN, I.-C., AND OUHYOUNG, M. 2005. Mirror mocap: Automatic and efficient capture of dense 3d facial motion parameters from video. *The Visual Computer 21*, 355–372.

MA, W.-C., JONES, A., CHIANG, J.-Y., HAWKINS, T., FREDERIKSEN, S., PEERS, P., VUKOVIC, M., OUHYOUNG, M., AND DEBEVEC, P. 2008. Facial performance synthesis using deformation-driven polynomial displacement maps. *ACM Trans. Graph. 27*, 121:1–121:10.

MADSEN, K., NIELSEN, H. B., AND TINGLEFF, O., 2004. Methods for non-linear least squares problems (2nd ed.).

PIGHIN, F., AND LEWIS, J. P. 2006. Performance-driven facial animation. In *ACM SIGGRAPH Course Notes*.

RUSINKIEWICZ, S., AND LEVOY, M. 2001. Efficient variants of the ICP algorithm. In *3DIM*, 145–152.

SARAGIH, J. M., LUCEY, S., AND COHN, J. F. 2011. Real-time avatar animation from a single image. In *FG*, 213–220.

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 399–405.

VALGAERTS, L., WU, C., BRUHN, A., SEIDEL, H.-P., AND THEOBALT, C. 2012. Lightweight binocular facial performance capture under uncontrolled lighting. *ACM Trans. Graph. 31*, 187:1–187:11.

VIOLA, P., AND JONES, M. 2001. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 511–518.

WEISE, T., LI, H., GOOL, L. V., AND PAULY, M. 2009. Face/off: Live facial puppetry. In *SCA*, 7–16.

WEISE, T., BOUAZIZ, S., LI, H., AND PAULY, M. 2011. Realtime performance-based facial animation. *ACM Trans. Graph. 30*, 77:1–77:10.

ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. M. 2004. Spacetime faces: high resolution capture for modeling and animation. *ACM Trans. Graph. 23*, 548–558.