Facial Performance Enhancement Using Dynamic Shape Space Analysis

AMIT H. BERMANO
ETH Zurich and Disney Research Zurich
DEREK BRADLEY
Disney Research Zurich
THABO BEELER and FABIO ZUND
ETH Zurich and Disney Research Zurich
DEREK NOWROUZEZAHRAI
Disney Research Zurich and Universite de Montreal
ILYA BARAN
Disney Research Zurich
OLGA SORKINE-HORNUNG
ETH Zurich
HANSPETER PFISTER
Harvard University
ROBERT W. SUMNER and BERND BICKEL
Disney Research Zurich
and
MARKUS GROSS
ETH Zurich and Disney Research Zurich

The facial performance of an individual is inherently rich in subtle deformation and timing details. Although these subtleties make the performance realistic and compelling, they often elude both motion capture and hand animation. We present a technique for adding fine-scale details and expressiveness to low-resolution art-directed facial performances, such as those created manually using a rig, via marker-based capture, by fitting a morphable model to a video, or through Kinect reconstruction using recent *faceshift* technology. We employ a high-resolution facial performance capture system to acquire a representative performance of an individual in which he or she explores the full range of facial expressiveness. From the captured data, our system extracts an expressiveness model that encodes subtle spatial and temporal deformation details specific to that particular individual. Once this model has been built, these details can be transferred to low-resolution art-directed performances. We demonstrate results on various forms of input; after our enhancement, the resulting animations exhibit the same nuances and fine spatial details as the captured performance, with optional temporal enhancement to match the dynamics of the actor. Finally, we show that our technique outperforms the current state-of-the-art in example-based facial animation.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*

General Terms: Algorithms

Additional Key Words and Phrases: Facial animation, performance enhancement, data-driven upsampling

ACM Reference Format:

Amit H. Bermano, Derek Bradley, Thabo Beeler, Fabio Zund, Derek Nowrouzezahrai, Ilya Baran, Olga Sorkine, Hanspeter Pfister, Robert W. Sumner, Bernd Bickel, and Markus Gross. 2014. Facial performance enhancement using dynamic shape space analysis. ACM Trans. Graph. 33, 2, Article 13 (March 2014), 12 pages.
DOI: http://dx.doi.org/10.1145/2546276

Authors' addresses: A. H. Bermano (corresponding author), ETH Zurich, Switzerland; email: amberman@ethz.ch; D. Bradley, Disney Research Zurich, Switzerland; T. Beeler, F. Zund, ETH Zurich, Switzerland; D. Nowrouzezahrai, I. Baran, Disney Research Zurich, Switzerland; O. Sorkine, ETH Zurich, Switzerland; H. Pfister, Harvard University, Cambridge, MA; R. W. Sumner, B. Bickel, Disney Research Zurich, Switzerland; M. Gross, ETH Zurich, Switzerland.

DOI: http://dx.doi.org/10.1145/2546276

1. INTRODUCTION

Facial expression plays a critical role in almost all aspects of human interaction and face-to-face communication. As such, realistic face modeling has long been considered a grand challenge in the field of computer graphics. Overcoming this challenge is especially difficult since human faces can accommodate such a large range of

expressiveness, from the most subtle hint of emotion to exaggerated exclamations. In real-life communication, subtle changes in facial deformation and dynamics can have a significant impact on the perceived expression and meaning conveyed by an individual. For example, a genuine smile may differ from a forced smile only in the slight tensing of one's cheeks. These subtle changes also contribute to the individuality that makes any particular person's face unique. Two different individuals may have a vastly different range and style of facial expressiveness.

A great deal of progress has been made toward solving this grand challenge, including sophisticated facial rigs, skin rendering algorithms, facial motion capture devices, and animation interfaces. However, despite these significant research contributions, creating synthetic facial performances that are as compelling and as expressive as a real actor's performance remains an elusive task. As the desired level of realism increases, animators must spend increasing amounts of time to incorporate the nuances of deformation that are characteristic of a particular actor's performance. At some point, the details become so subtle that they even elude the most skilled animators. Facial motion capture techniques based on marker-tracking, depth cameras like the Kinect, and fitting parametric models to video, also have a limited spatial and temporal resolution. Fine-scale details may escape the fidelity of the capture technology, especially when head-mounted devices are required. These missing details contribute to an unfortunate result: many attempts at realistic facial animation fall prey to the "uncanny valley" effect and are perceived as eerie and lifeless.

Our work contributes to realistic facial animation by targeting the subtle details of deformation and timing that escape both hand animation and motion capture systems and render an individual's facial performance unique and compelling. To this end, we propose a novel data-driven technique to enhance the expressiveness of facial geometry and motion. We start by recording a highly detailed representative performance of an individual in which he or she explores the full range of facial expressiveness. From this data, our system extracts a model of expressiveness that encodes the subtleties of deformation specific to that individual. Once built, this model is used to automatically transfer these subtleties to lower-resolution facial animations that lack expressive details. The input animation will be augmented with the subtle deformations particular to the individual's face, increasing the perceived expressiveness and realism. Our system also takes advantage of the timing information in our database by enhancing facial keyframe interpolation so that the nonlinearities of expression formation exhibited by the real actor's performance are reflected in the interpolated and enhanced result. We demonstrate the robustness of our approach by enhancing a variety of input animations, including hand-animated facial rigs, face models driven by low-resolution motion capture data, morphable models animated using video data, and performance reconstructions generated with a Kinect using recent *faceshift* technology[1]. After our enhancement, the resulting animations exhibit the nuances and fine details of the original performance. Finally, we show that our algorithm outperforms the current state-of-the-art approach for data-driven facial performance synthesis [Ma et al. 2008].

In summary, our main contributions are:

—a framework for data-driven spatial enhancement of low-resolution facial animations, using a compressed shape space;

—a novel method for enhancing facial keyframe interpolation of temporal performances;

—validation of our framework on four different types of input facial animations, with a direct comparison to state-of-the-art.

[1]www.faceshift.com.

2. RELATED WORK

3D Face Scanning. There has been a lot of work on capturing the geometry of human faces. Some methods focus on capturing high-resolution static poses [Weyrich et al. 2006; Ma et al. 2007; Beeler et al. 2010] that can then be animated with marker-based motion capture data [Williams 1990]. Others use space-time stereo to capture low-resolution 3D models at interactive rates [Borshukov et al. 2003; Wang et al. 2004; Zhang et al. 2004; Zhang and Huang 2006]. Neither approach is capable of simultaneously capturing high-resolution spatial and temporal details. Bickel et al. [2007] combine high-resolution static geometry with motion capture data for large-scale deformations and add medium-scale expression wrinkles tracked in video. Huang et al. [2011a] leverage motion capture and static 3D scanning for facial performance acquisition.

Recent approaches use high-speed cameras and photometric stereo to capture performance geometry [Wenger et al. 2005; Jones et al. 2006; Ma et al. 2008]. Some of these techniques use time-multiplexed illumination patterns and consequently require an acquisition rate that is a multiple of the final capture rate. Wilson et al. [2010] introduce a temporal upsampling method to propagate dense stereo correspondences between frames to reconstruct high-resolution geometry for every captured frame. Bradley et al. [2010] use a completely passive system with high-resolution cameras. We use an extended version of the passive system by Beeler et al. [2011] to capture dynamic high-resolution 3D geometry for our expression database. The specifics of how this database is captured are not important, and a number of alternative methods could be used for this purpose.

Facial Animation. Facial animation has a long history that goes back to the early '70s [Parke 1974]. Some methods use models of facial anatomy [Terzopoulus and Waters 1993] that can be combined with physical models of skin deformation [Wu et al. 1996; Sifakis et al. 2005; Venkataraman et al. 2005; Zhang and Sim 2005]. Another approach is to use deformable 3D face models [Blanz et al. 2003; Vlasic et al. 2005] and fitting them to video data [Li et al. 1993; Essa et al. 1996; DeCarlo and Metaxas 1996; Pighin et al. 1999; Dale et al. 2011]. Methods based on example poses and shape interpolation (i.e., blendshapes) [Lewis et al. 2000, 2005: Chuang and Bregler 2002; Seol et al. 2011] are especially popular in the entertainment industry because of their intuitive and flexible controls and can even be driven in real time from video [Chai et al. 2003] or the Microsoft Kinect device [Weise et al. 2011]. Similar concepts can also be applied to drive a set of hand-drawn faces for generating performance-driven, "hand-drawn" animation in real time [Buck et al. 2000]. None of these approaches reaches the quality of high-resolution performance-driven facial animation from person-specific captured data, and animation of subtle facial details and dynamics are still elusive. Our approach tries to bridge the gap between traditional facial animation and high-quality 3D face scanning.

Deformation and Detail Transfer. Static geometry can be enhanced with details transferred from different models by means of simple displacements (for small detail) or differential coordinates (for substantial enhancements) [Sorkine et al. 2004; Takayama et al. 2011]. With such methods, the transferred detail is explicitly given, rather than being a function of the low-resolution pose. Deformation

transfer techniques [Sumner and Popović 2004] such as expression cloning [Noh and Neumann 2001; Pyun et al. 2003] transfer vertex displacements or deformation gradients from a source face model to a target face model with possibly different geometry. Similarly, data-driven approaches (e.g., based on canonical correlation analysis [Feng et al. 2008] or Gaussian process models [Ma et al. 2009]) learn and transfer facial styles. These techniques are typically applied to low-resolution geometry or low-frequency deformations. Golovinskiy et al. [2006] add static pore detail from a database of high-resolution face scans using texture synthesis. Huang et al. [2011b] train a collection of mappings defined over regions locally in both the geometry and the pose space for detailed hand animation. Bickel et al. [2008] use radial basis functions to interpolate medium-scale wrinkles during facial performance synthesis and transfer. Ma et al. [2008] add high-resolution facial details to a new performance using a compressed representation of vertex displacements. Notably, Alexander et al. [2010] use high-resolution scans to generate a detailed blendshape rig. In contrast to these methods, we present a framework that enables both spatial and temporal performance enhancement, which can be applied to various forms of art-directed facial animation, augmenting the high-resolution details and matching the dynamics of the particular individual's face.

Temporal Performance Synthesis. The temporal aspects of facial performances are very important for synthesis of new facial animations from speech [Bregler et al. 1997; Brand 1999; Ezzat et al. 2002; Kshirsagar and Thalmann 2003; Cao et al. 2004; Ma et al. 2004; Deng et al. 2005]. Most of these approaches record facial motion of speaking subjects and then recombine the recorded facial motion from learned parametric models to synthesize new facial motion. Chai and Hodgins [2007] learn a statistical dynamic model from motion capture data and generate animations from user-defined constraints solving a trajectory optimization problem. However, none of these methods takes high-resolution spatial details into account. Instead of learning a model we use a simpler and more general data-driven approach for performance synthesis of temporal and spatial details. Note that we do not specifically target temporal enhancement of speech animation.

3. OVERVIEW

Our goal is to enhance low-resolution facial performances by adding subtle facial features such as small wrinkles and pores, and/or temporal retiming to match the dynamics of a real actor. The resulting animations should respect the underlying artistic content and enhance the expressiveness of the intended performance. This is achieved through the use of a high-resolution temporally coherent performance database, as illustrated in Figure 1.

Preprocessing. We intend to process low-resolution animation sequences that contain the creative intent of the animator or actor, but lack facial expressiveness and fine-scale details. In order to enhance the details, for a given actor, we build a dense performance capture database $\mathcal{D}$, consisting of $|\mathcal{D}| \approx 1000$ frames of facial geometry with consistent connectivity, captured using a high-resolution (e.g., pore-level detail) performance capture technique [Beeler et al. 2011]. The database $\mathcal{D}$ is encoded into a *shape space*, which enables matching, projection, and interpolation. This shape space is defined using the polar decompositions of the deformation gradients [Sumner and Popović 2004] with respect to a neutral frame $\mathbf{d}_0$, and thus effectively represents the stretching and rotation of each triangle (Section 4).

Spatial Performance Enhancement. Given an input animation $\mathcal{A}$, our enhancement algorithm (Section 5) combines its low-frequency components with the high-frequency components of corresponding frames from $\mathcal{D}$. For performance enhancement, each input frame is projected onto the shape space spanned by $\mathcal{D}$, and the relevant high-frequency components are interpolated. These high-frequency details are then composed with their low-frequency counterparts, originating from the input frame, to generate the augmented mesh. The result is an upsampled version of the input animation, retaining the art-directed performance but enhanced with actor-specific expressiveness and details.

Temporal Performance Enhancement. Often, art-directed facial animations are created by hand, for example, using a facial rig. In this case, it is common practice to represent the animation as a set of keyframes and then interpolate the in-between frames. Unfortunately, this interpolation may not match the true dynamics of the real actor, resulting in an unrealistic performance. As an added benefit of our performance enhancement system, we can augment the temporal component of the performance in a data-driven manner (Section 6). Keeping the artist in the loop when defining keyframes, we devise a new interpolation scheme to retime the animation according to the actor-specific dynamics encoded in the database.

4. PREPROCESSING

Several preprocessing steps can be performed once per actor. The database must be constructed (Section 4.1), and encoded into our shape space (Section 4.2), in a region-based manner (Section 4.3).

4.1 Performance Capture Database

We acquire a dense database of detailed facial geometry that includes pores, wrinkles, and expressive deformations. The actor performs a number of short but expressive sequences that are stored in the database $\mathcal{D}$. The high-resolution geometry must be in full correspondence over time so that the motion and deformation of every point on the face is known. The database can be acquired using any high-resolution 3D facial performance capture method and we employ the passive approach of Beeler et al. [2011]. In this method, multiview video sequences are recorded and high-resolution per-frame geometry is computed with the static reconstruction method of Beeler at al. [2010] at approximately 40 frames per second. We then temporally align multiple sequences using dense image-space matching and per-frame geometry propagation, yielding a temporally consistent database of 24 expressive performances (in full vertex correspondence). Figure 2 shows a subset of poses from the database for each of our two actors.

4.2 Data Encoding

Frequency Separation. Once the database is captured, we separate the low- and high-frequency components of $\mathcal{D}$ using a low-pass filter operation $f(\cdot)$. We create the dataset $f(\mathcal{D})$, where $f(\mathcal{M})$ for a set of meshes. $\mathcal{M}$ denotes the set of all its filtered meshes, that is, $f(\mathcal{M}) = \{f(\mathbf{m}) | \mathbf{m} \in \mathcal{M}\}$. In our work, this separation is conducted using implicit curvature flow [Desbrun et al. 1999], an iterative approach where in each iteration we find new vertex positions X^{n+1} by solving the system

$$(I - \lambda dt K)X^{n+1} = X^n,$$
$$K_{ij} = \begin{cases} -\frac{1}{4A_i}(cot\alpha_j + cot\beta_j) & i \neq j, \\ -\sum_{k \neq i} K_{ik} & i = j \end{cases} \quad (1)$$

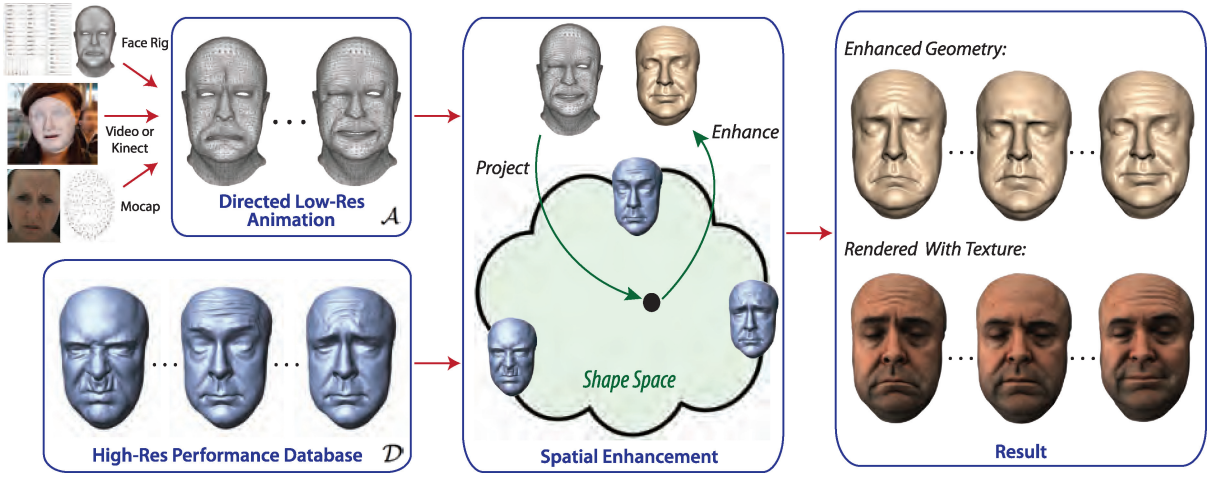

Fig. 1. Our spatial enhancement approach takes as input a low-resolution animation and a high-resolution performance database, and enhances the input animation with actor-specific facial details.

Fig. 2. Performance capture database samples for our two actors.

where α_j and β_j are the two angles opposite to the edge in the two triangles having the edge e_{ij} in common, A_i is the sum of the areas of the triangles having x_i as a common vertex, and λdt was chosen to be 125 in all our experiments. By using a large λdt, we require only a single iteration to aggressively attenuate the high-frequency components while preserving roughly the same levels for the low-frequency part. During performance enhancement we will also separate the frequencies of the input animation (Section 5.1 and Figure 5), so this step creates a common ground for all different types of inputs presented in this article and enables an accurate matching and enhancement process. Note that this process preserves the size of input triangles, which might yield very small or nearly degenerated ones. To avoid the numerical instabilities caused by such triangles, we also perform one iteration of uniform Laplacian smoothing after the implicit fairing process.

Encoding. Finally, the database frames are encoded into the shape space. For every database frame $\mathbf{d} \in \mathcal{D}$, the deformation gradients encoding the difference between the mesh and its lowfrequency counterpart $f(\mathbf{d}) \in f(\mathcal{D})$ are encoded and saved, denoted by $\mathbf{d}^h$. These high frequencies will be used as details, and are transferred onto the input animations during the spatial enhancement process (Section 5.4).

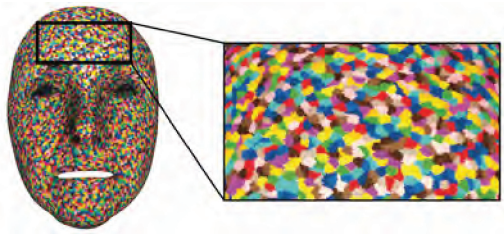

Fig. 3. Triangle clustering for encoding into a *compressed* shape space. We encode the average deformation gradient for each patch rather than per-triangle deformation gradients.

For the projection, we encode the low-frequency component relative to the low-frequency neutral pose $f(\mathbf{d}_0)$. Since the input is fairly low resolution, the full shape space of the high-resolution mesh contains redundant information. To reduce the runtime and memory footprint, we encode into a *compressed* shape space. Rather than encoding the deformation gradient per triangle, we uniformly cluster the high-resolution mesh into patches and encode only the average deformation gradient for each patch. Patches are computed using a random seed-and-grow approach. In practice, we found that patches of 100 vertices provided ample compression for our high-resolution meshes. The shape space vector size is thus reduced by a factor of 100. Figure 3 illustrates the clustering on one of our datasets. We denote the set of all compressed shape space vectors in our database as $\tilde{\mathcal{D}}$. Note that alternative compression schemes are possible, for example, mesh simplification. However, deformation gradients would have to be computed a second time on the simplified mesh, and therefore we use the clustering approach.

4.3 Regions

Tena et al. [2011] show that region-based face models generalize better than their holistic counterparts. Regions are a partition of the

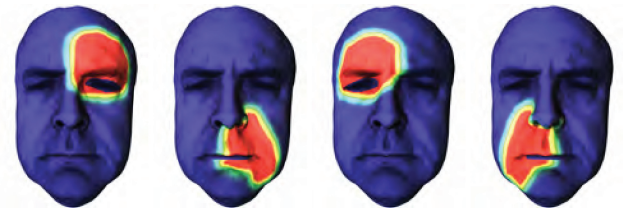

Fig. 4. The four voluntary regions that are used in the matching process. The weights smoothly decrease from 1 (red) to 0 (blue).

mesh faces into what are usually groups with common functionality. The regions $\mathcal{R}$ we have chosen to use are based on Tena et al.'s work, which clusters the vertices according to their correlation in movement. While using the same clustering, we distinguish between *voluntary regions*, controlled by the actor directly, and *involuntary regions*. The latter are areas that deform indirectly, governed by the voluntary regions. As shown in Figure 4, out of the total 13 regions described by Tena and colleagues, we classify four as voluntary and use them in our matching process: left and right halves of the mouth, and the left and right eyebrows. This classification enables us to detect asymmetrical expressions as well as decoupling of the eyes and mouth. Note, however, that as elaborated in Section 5.3 and decipted in Figure 4, this is a soft-boundary decoupling; each mesh triangle is weighted according to its geodesic distance from each of the regions. In Section 5.4, these regions will be used both for matching and blending of the high-frequency details originating from several database meshes.

5. PERFORMANCE ENHANCEMENT MODEL

Our data-driven performance enhancement approach consists of five steps for each frame. First, the frame is brought into correspondence with the database geometry and we perform frequency separation (Section 5.1). Next, the frame is *encoded* into the shape space (Section 5.2). Then, we project it onto the shape space of the database in a *matching* step (Section 5.3). Based on the matching, we *interpolate* the relevant high-frequency details from the database and compose them with the low-frequency input mesh (Section 5.4). Finally, we *reconstruct* the resulting mesh, and assign per-vertex colors to it by linearly interpolating the colors from the same database frames (Section 5.5).

5.1 Input Animation Pre-Processing

Input Sources. The input animations $\mathcal{A}$ can come from any source, but in industry these are most often created using a manually controlled rig or driven by sparse marker-based motion capture. For one of our experiments, we use a facial rig as input built from a set of $B \approx 40$ blendshapes. These are based on static scans of an actor according to the Facial Action Coding System (FACS) [Ekman and Friesen 1978]. This rig can be fully controlled manually, allowing artists to create arbitrary animations. The rig only spans an approximate subset of facial expressions, and there is a natural limit on the accuracy and number of animation parameters an animator can evolve over time. Some example snapshots from an animation created using this facial rig are shown in Figure 10 (left column).

Additionally, we evaluate our algorithm on three other input sources, including sparse marker-based motion capture data, a low-resolution morphable model fit to a monocular video sequence [Dale et al. 2011], and a blendshape-based facial animation driven by Kinect data using *faceshift* [Weise et al. 2011].

Registration and Frequency Separation. The input animation $\mathcal{A}$ will have a different geometric structure than $\mathcal{D}$ (e.g., it could be a lower-resolution mesh or possibly just marker positions). In our examples the actor is the same for $\mathcal{A}$ and $\mathcal{D}$, although this need not be the case. In theory, we could transfer facial details to different actors, although in practice, the facial properties of the face in the database $\mathcal{D}$ and the animation $\mathcal{A}$ should be similar to achieve visually plausible results.

To be able to work in the same shape space and to provide a means for detail transfer, we obtain a dense correspondence between a database neutral frame $\mathbf{d}_0$ and the neutral frame $\mathbf{a}_0$ from our low-resolution input source. We start by aligning $\mathbf{a}_0$ to $\mathbf{d}_0$ using the nonrigid registration method of Li et al. [2008]. This establishes a correspondence between the database meshes and the input data that we can propagate over the entire input. Naturally, the number of vertices n_d for a pose in $\mathcal{D}$ is much larger than the number of vertices n_a for a pose in $\mathcal{A}$. For example, in the case of the face rig, $n_a \approx 4000$, whereas $n_d \approx 1.2M$. We therefore employ a linear deformation model to propagate the animation specified by $\mathcal{A}$ to the registered high-resolution neutral pose [Bickel et al. 2008]. The resulting animation $\bar{\mathcal{A}}$ matches exactly the motion of $\mathcal{A}$ and is in dense correspondence with our database $\mathcal{D}$. We then separate the low frequencies in $\bar{\mathcal{A}}$ using the same procedure as with the database (Section 4.2), to obtain $f(\bar{\mathcal{A}})$. $f(\bar{\mathcal{A}})$ is now a standard form that is similar to $f(\mathcal{D})$, no matter the source of the original animation $\mathcal{A}$. The results of this process are decipted in Figure 5.

5.2 Encoding

As mentioned in Section 3, our performance enhancement uses deformation gradients [Sumner and Popović 2004]. In order to achieve accurate matching, every smoothed input frame $f(\tilde{\mathbf{a}}) \in f(\bar{\mathcal{A}})$ is first rigidly aligned to the database using the method of Horn [1987], and only then is the frame encoded. The deformation gradients are encoded with respect to a smoothed version of a database neutral pose, into a vector denoted $\mathbf{a}^{\ell}$. As explained in Section 4.2, a compressed version $\tilde{\mathbf{a}}$ of the same vector is created to be used in Section 5.3. The rotation $\mathbf{R}_a$ and translation $\mathbf{T}_a$ matrices that are produced during the alignment operation are also stored along with the deformation gradients vector, and all are used for reconstruction in Section 5.5.

5.3 Matching

In order to transfer the subtleties of facial details recorded in our database to a low-resolution input mesh, we must locate the corresponding high-frequency data in our database. This can be formulated as a projection of the compressed shape space vector $\tilde{\mathbf{a}}$ of the input frame onto the database. In other words, we represent the input frame as a convex combination of weights $\mathbf{w}$ that represents a point in the shape space, spanned by the database, that is closest to the input frame, namely

$$\min_{\mathbf{w}} \|(\tilde{\mathbf{D}} \cdot \mathbf{w}) - \tilde{\mathbf{a}})\|, s.t. \sum_{w_i \in \mathbf{w}} w_i = 1, w_i > 0, \qquad (2)$$

where each column i in the matrix $\tilde{\mathbf{D}}$ represents the compressed shape space vector of frame i in our database.

Previous works have restricted this matching to affine weights [Baran et al. 2009]. However, in our experiments, affine weights distorted fine features such as pores and we therefore enforce convex weights, solving the resulting minimization problem

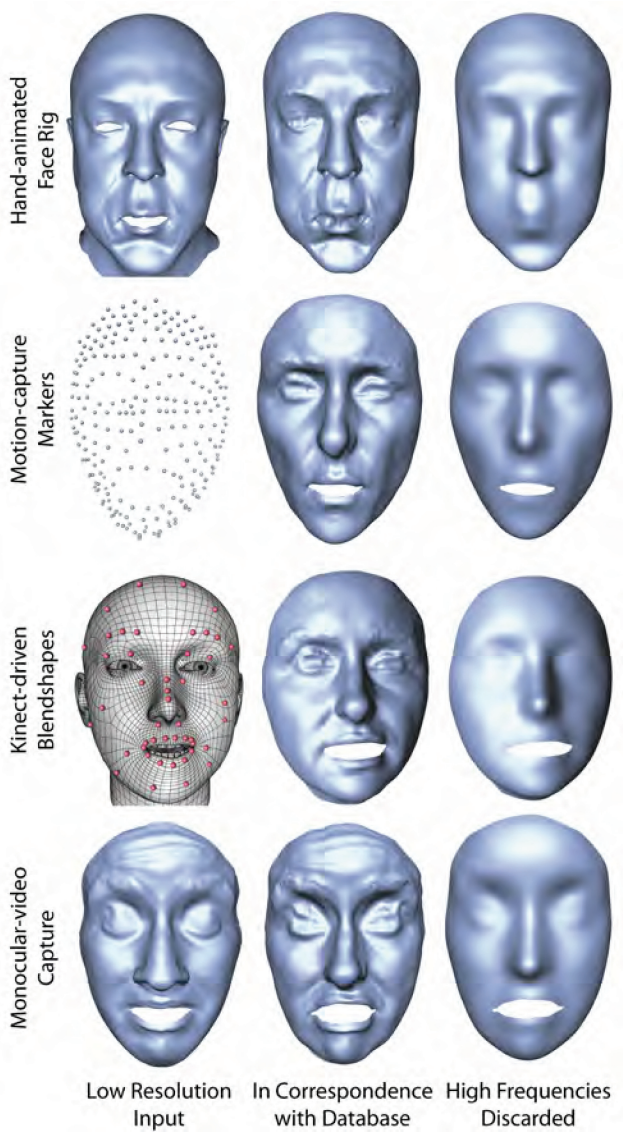

Fig. 5. Input frame pre-processing for all input sources (top to bottom): Hand-animated rig, tracked mocap markers, depth-camera-driven rig and monocular video-based capture. The input frame (left) drives a deformation of the database neutral pose (middle) and is standardized using smoothing (right).

using a QP solver. Note that the weights quickly fall off to nearly zero outside the immediate neighborhood of $\tilde{\mathbf{a}}$, yielding only a small number of relevant shapes to interpolate. Furthermore, in Section 4.3 we describe a partition of the face into *voluntary regions*, $\mathcal{R}$. This implies that throughout the matching process each region of the face is treated independently. However, actual facial expressions are not decoupled between regions, as a genuine smile is shown on the eyes as well as on the mouth. Therefore, in our method, each region is represented as a vector ω_r of weights per mesh triangle. The triangle weights are constant within the region, and decay in a Gaussian way as the geodesic distance from this area grows. This means that, with diminishing influence, areas outside the region participate in the matching process, yielding a subtle coupling effect. We incorporate these weights in a weighted least squares manner, solving

$$(\tilde{\mathbf{D}}^T \cdot diag(\omega_r) \cdot \tilde{\mathbf{D}})\, \mathbf{w}^r = \tilde{\mathbf{D}}^T \cdot diag(\omega_r)\, \tilde{\mathbf{a}} \qquad (3)$$

$$\forall r \in \mathcal{R},$$

with the aforementioned convex constraints, where $diag(\mathbf{v})$ is the diagonal matrix with $\mathbf{v}$ on its *diag*onal. In order to save runtime and memory consumption, the matrices $(\tilde{\mathbf{D}}^T \cdot diag(\omega_r) \cdot \tilde{\mathbf{D}})$ are precomputed per region, and only $\tilde{\mathbf{D}}^T \cdot diag(\omega_r) \cdot \tilde{\mathbf{a}}$ are computed during the matching process.

5.4 Interpolation

Having computed the weights per region, we are now able to generate the highly detailed augmented mesh. The high-frequency details of the database are linearly blended according to the computed weights with respect to the region weights ω_r. Formally, given a triangle t and a component i of its deformation gradient, the interpolation scheme is

$$\mathbf{b}^h = \sum_{r=0}^{|\mathcal{R}|} \mathbf{D}^h_{t_i}\, \mathbf{w}^r\, \frac{\omega_{r,t}}{\sum_{\rho=0}^{|\mathcal{R}|} \omega_{\rho,t}}, \qquad (4)$$

where $\mathbf{D}^h$ consists of columns which are the high-frequency detail vectors $\mathbf{d}^h$, $\mathbf{D}^h_{t_i}$ is the row in $\mathbf{D}^h$ that corresponds to the i-th component of the deformation gradient of triangle t, and $\omega_{r,t}$ is the t-th element in the vector ω_r. Note that, per region r, the length of the matched weights vector $\mathbf{w}^r$ is $|\mathcal{D}|$ while the length of ω_r is the number of triangles in a mesh. As mentioned before, this interpolation scheme blends the database within each region according to the matched weights and provides a normalized interpolation between regions. As a final step before reconstruction, the blended high-frequency details $\mathbf{b}^h$ are composed with the low-frequency deformation gradients of the input animation $\mathbf{a}^\ell$. This is done by converting the stretching and rotation vectors of $\mathbf{b}^h$ and $\mathbf{a}^\ell$ back to the deformation gradients' matrices and multiplying them. This process is equivalent to applying the blended deformation gradients representing the high-frequency details to the smoothed input animation mesh, as a deformation transfer [Sumner and Popović 2004], only without explicitly producing the intermediate low-frequency mesh.

5.5 Reconstruction

Having the final deformation gradients, we reconstruct the mesh using a slightly modified version of the Laplace-Beltrami operator [Botsch et al. 2006]. First, in the interest of runtime performance, the Laplace-Beltrami operator is considered to be similar for all meshes, and so it is precomputed and prefactored once for the neutral pose. This assumption allows us to only use back-substitutions during reconstruction and has proven to be reasonable in all our experiments. Second, since the final deformation gradients are composed from several different ones, some artifacts tend to appear along the mesh boundaries. To suppress this artifact, we add a weighted regularization term to the reconstruction system of equations: in addition to the Laplace-Beltrami operator, we minimize the 1D Laplacian term along the mesh boundaries. In all our experiments, a minimal weight factor of $w = 0.05$ was sufficient to completely eliminate the artifacts.

After reconstructing the final mesh, we align it to the database using the method of Horn [1987] and we then apply the inverse transformations $\mathbf{R}_a^{-1}$ and $\mathbf{T}_a^{-1}$ that were calculated during encoding to restore the mesh to its starting position.

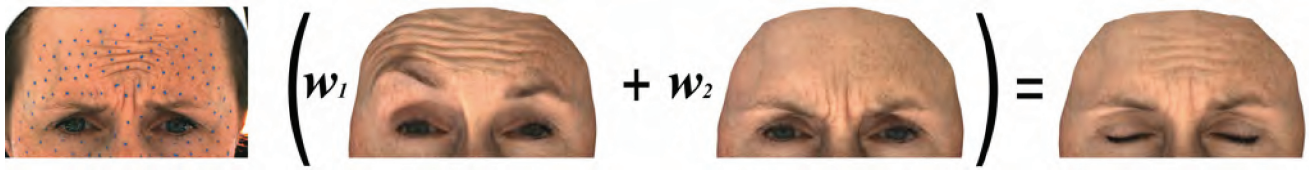

Fig. 6. To create the desired forehead wrinkles on the left, which are not present in the database, our enhancement technique blends between two different database frames.

As a final step, we perform the same interpolation scheme described in Section 5.4 on the vertex colors of the database meshes, and apply the result to the final mesh. An illustration of our enhancement technique is shown in Figure 6 for the forehead regions. Since the given expression is not in the database, two database frames are blended to create the closest match.

6. TEMPORAL PERFORMANCE ENHANCEMENT

In facial animation, the dynamic behavior of a performance greatly affects its perceived realism. Often, correct dynamics can be difficult to achieve. For example, when an animator creates a facial animation by rigging keyframes, the keyframes are interpolated linearly or with some hand-adjusted ease-in/ease-out curves to create the full animation. This simple interpolation is insufficient to capture the timing of a real performance, and affects realism. In this section, we describe a method to automatically adjust the temporal behavior of the keyframe interpolation in a data-driven manner, using the previously described capture database.

The core concept that enables the temporal performance enhancement is the extension of the previously described frame projection to a sequence projection operator (Section 6.1). In short, given two sequences, this operator determines how one of the sequences can be approximated by the other, and how close the approximation.

To start the process, the artist picks two keyframes he or she wishes to interpolate, as well as the length of the desired motion. These two keyframes are then treated as a sequence of length two, and are projected onto our compressed shape space using the sequence projection operator to find the closest matching sequence. The temporal behavior of the matched sequence is analyzed, and this information is used to generate a well-timed interpolation of the input frames (Section 6.2). The result is an animation that closely follows the data-driven dynamic behavior while maintaining the user's artistic intent and spatial features. Note that the process is invoked by request of the artist, since temporal performance modification may not be desired in every scenario.

6.1 Sequence Projection

Sequence projection is an extension of the single frame projection operator described in Section 5.3. Given an input sequence, we wish to find the closest sequence in the database. This is a nontrivial task since the sequences in the database can have different dynamics and temporal behavior, resulting in different timing. For the sake of simplicity, we disregard the partitioning of the face into regions in this explanation, although the method is applied to each region independently.

As mentioned in Section 4.1, our capture database consists of sequences transitioning from the neutral pose to an extremity and back. We consider each such sequence as a temporal continuum, sampled uniformly at the sequence frames. The task of projecting an input sequence, α, onto such a sequence in the database, β, is simply one of finding a *valid* mapping between each frame of α to the timeline defined by β. A valid mapping is one that preserves the temporal order: a later frame in the input sequence α must be projected onto a later point in the timeline defined by the database sequence β.

Given an input sequence α, consisting of m frames $\{\alpha_0 \dots \alpha_{m-1}\}$, and a database sequence β, consisting of n frames $\{\beta_0 \dots \beta_{n-1}\}$, we start by projecting each frame α_i onto each of the linear segments $\{\beta_j, \beta_{j+1}\} \subset \beta$, and store the resulting blend weights as a matrix $\mathbf{T}$, as well as the distance (or error) of the projected points from the original ones as a matrix $\mathbf{E}$.

$$\begin{aligned} \mathbf{T}_{i,j} &= \text{Proj}(\alpha_i, \{\beta_j, \beta_{j+1}\})_0, \qquad 0 \le i < m,\ 0 \le j < n-1 \\ \mathbf{E}_{i,j} &= \|\alpha_i - (\mathbf{T}_{i,j}\beta_j + (1 - \mathbf{T}_{i,j})\beta_{j+1})\| \end{aligned} \tag{5}$$

Here $\text{Proj}(\mathbf{v}, \mathcal{S})$ is a vector of blend weights that represents the static projection (Section 5.3) of vector $\mathbf{v}$ on the set $\mathcal{S}$, and $\text{Proj}(\mathbf{v}, \mathcal{S})_0$ is the first element of this vector.

Next, we wish to identify the *valid* mapping P that yields the minimum error. This means that we search for a monotonic function that assigns a segment $\{\beta_j, \beta_{j+1}\}$ to every input frame α_i in a valid way, and minimizes the sum of projected distances. We solve for P that minimizes the following objective.

$$\begin{aligned} &\min_P \sum_{i=0}^{m-1} \mathbf{E}_{i,P(i)} \\ &s.t.\ P(i_1) \le P(i_2) \qquad \forall i_1 < i_2. \end{aligned} \tag{6}$$

We solve this minimization problem using dynamic programming. This process is performed for all the sequences in the database, and the resulting projected sequence corresponding to the input is chosen to be the one yielding the minimal error. Note that in most cases the input sequence α is projected only to a part of the chosen database sequence, which we refer to as the projected *subsequence*.

6.2 Temporally Driven Interpolation

We now describe how we use the sequence projection operator to interpolate the selected keyframes in a data-driven manner. As mentioned earlier, the user selects the desired length of the resulting sequence, m, and two endpoint keyframes of the resulting sequence, α_0 and α_{m-1}. As a first step, the sequence (of length two) $\{\alpha_0, \alpha_{m-1}\}$ is projected onto the database to find the closest subsequence. In case the input matches only a portion of a database sequence, the user may choose to *extend* the matched subsequence to the full corresponding database motion.

The matched sequence $\beta = \{\beta_0 \dots \beta_{n-1}\}$ has exactly the dynamics we want, but typically contains a different number of frames n than the desired m. To rectify this, we start by creating a new sequence $\hat{\beta} = \{\hat{\beta}_0 \dots \hat{\beta}_{m-1}\}$, which is a uniform interpolation of the ends of the selected subsequence $\{\beta_0, \beta_{n-1}\}$. Then, to get the right dynamics we wish to position each frame $\hat{\beta}_i$ on the continuous timeline uniformly sampled by β. This is accomplished by projecting $\hat{\beta}$

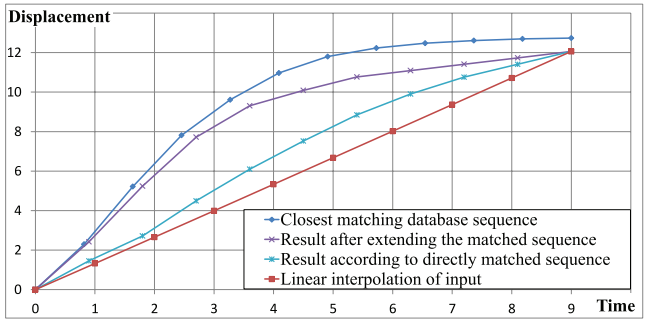

Fig. 7. A transition between the rest pose and a smile is enhanced using the temporal enhancement method. Here we show linear interpolation of one vertex (red), temporally augmented interpolation according to the closest matched subsequence (cyan), and temporally augmented interpolation according to the extended full sequence (purple). The actual temporal behavior of the database expression is presented for reference (blue).

onto β, again using the sequence projection operator. If a frame $\hat{\beta}_i$ is mapped to the segment $\{\beta_j, \beta_{j+1}\}$ with blend weight t_i, one could deduce that $\hat{\beta}_i$ is projected to the point $j+(1-t_i)$ in the continuous timeline. These timestamps are recorded for each frame.

As a final step, the two input keyframes α_0, α_{m-1} are linearly interpolated, creating the sequence $\hat{\alpha} = \{\hat{\alpha}_0 \dots \hat{\alpha}_{m-1}\}$. The frames are assigned the previously computed timestamps $t(\hat{\beta}_i)$, forming a nonuniformly sampled piecewise linear time curve. The curve is then resampled using the shape space, in uniform intervals of $dt = n/m$ for the final animation. The result is a sequence composed solely of the artistically generated keyframes, but with the temporal behavior of the matched sequence of the database.

Figure 7 illustrates the result of the retiming process, performed on a rigged transition between the rest pose and a smile. In this figure, we plot the time versus displacement of a vertex positioned on the edge of the mouth. An input linear interpolation is shown in red, and the closest matching database sequence is shown in dark blue. The database sequence contains the nonlinear dynamics of the actor, but it represents a larger smile, that is, the amount of displacement extends further than the input keyframe. The input sequence directly matches the first part of the database smile, where the motion is fast and relatively linear. If we retime the input sequence using this direct match, we obtain the subsequence behavior shown in cyan, which is not much of an enhancement. However, should the artist choose to retime the input after extending the matched subsequence to the full database smile, we obtain temporal behavior that matches the captured data as closely as possible while maintaining the artistic intent (shown in purple).

The accompanying video exhibits a rigged transition of two keyframes, after being temporally and spatially enhanced. In order to keep the full integrity of the artistic intent, we propose to perform temporal enhancement before the spatial enhancement described in Section 5 is applied, and to enhance between only two interpolated keyframes at a time, although these are not constraints of the method.

7. RESULTS

Our dynamic performance enhancement algorithm increases realism in facial animations by adding fine-scale details and expressiveness to low-resolution performances. In order to validate our technique, we captured a high-resolution performance of an actor with the same reconstruction technique that we used to build the expression database. The performance is then spatially downsampled to mimic a typical input that we would expect, lacking expressive details. We then enhance the performance using our technique, yielding a result very similar in detail to the ground-truth input scans. Figure 8 shows a few frames of the resulting validation. Minor differences between the ground-truth and enhanced meshes (for example, around the mouth) are only visible where the correct shape is simply not in the database. For all examples in this section, we invite the reader to refer to the accompanying video for more results.

Fig. 8. Validation of our performance enhancement on a downsampled high-resolution performance. Left: select frames from the high-resolution scan. Middle: downsampled inputs to the algorithm. Right: our enhanced result very closely matches the original.

We demonstrate the robustness and flexibility of our enhancement algorithm by augmenting facial performances generated using four radically different facial animation techniques commonly used in industry and research. First, in Figure 9, we enhance a traditional marker-based motion capture animation. Approximately 250 markers are tracked and used to drive a low-resolution facial animation. The traditional motion capture approach is to deform a face mesh (e.g., using a linear shell model) with the marker positions as constraints. As a result, fine-scale details are clearly missing since they cannot be reconstructed from such a sparse set of markers (Figure 9, third column). Our technique is able to enhance the result with detailed wrinkles (Figure 9, fourth column), greatly adding to the expressiveness of the performance. We also illustrate the result rendered with per-frame reconstructed textures (Figure 9, last column). Note that we purposely do not target eye motion in the enhancement algorithm, and so we choose a single capture frame with closed eyes and blend the eye regions into all final results using our interpolation framework.

Another common art-directable facial animation approach is a hand-animated rig. An example rigged performance and our enhanced result are shown in Figure 10. Most rig animations also lack fine-scale expression details, as it is time consuming and difficult for animators to author these subtle effects. Our enhancement approach successfully adds the high-frequency details automatically. To illustrate the result of shape space matching, Figure 11 shows some of the closest database poses that are used in the region-based interpolation for one of the rig result frames.

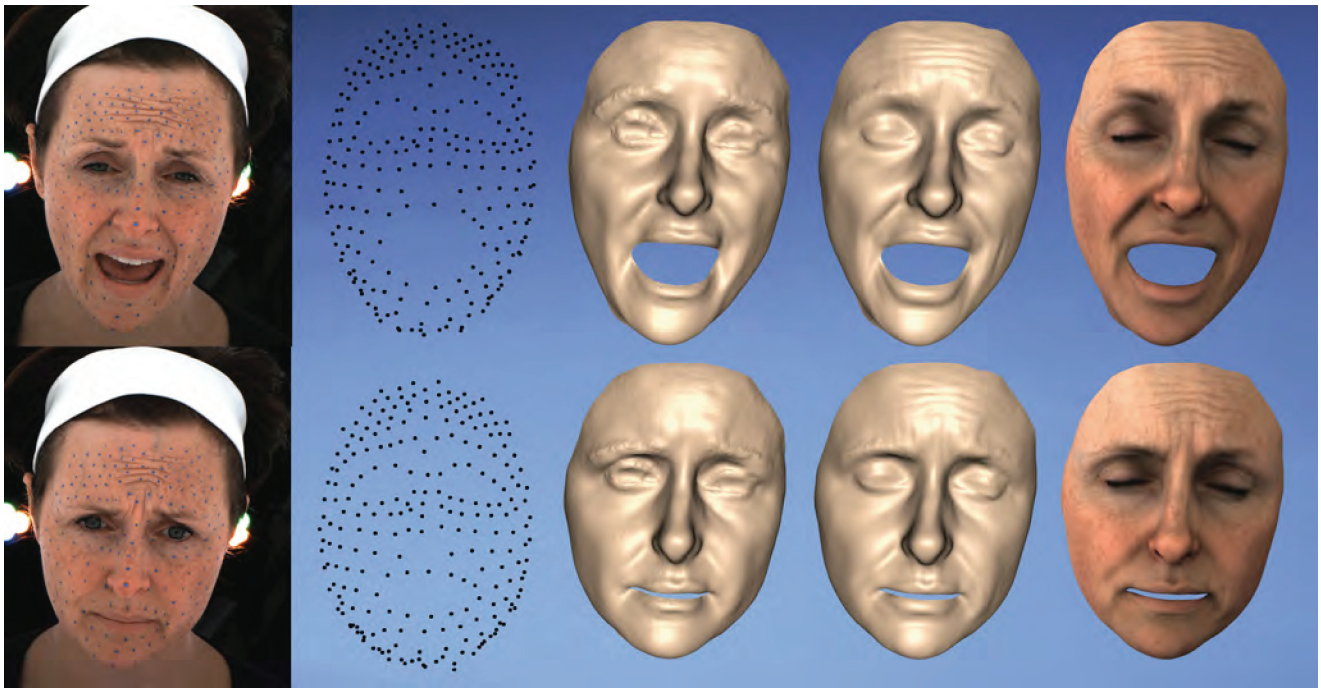

Fig. 9. Enhancing a marker-based motion-captured performance. The columns from left to right: selected frames from the input sequence, tracked marker positions, traditional mocap result using the tracked markers to deform the mesh with a linear shell (notice the missing expression wrinkles), our enhanced geometry including expression details, final result rendered with texture.

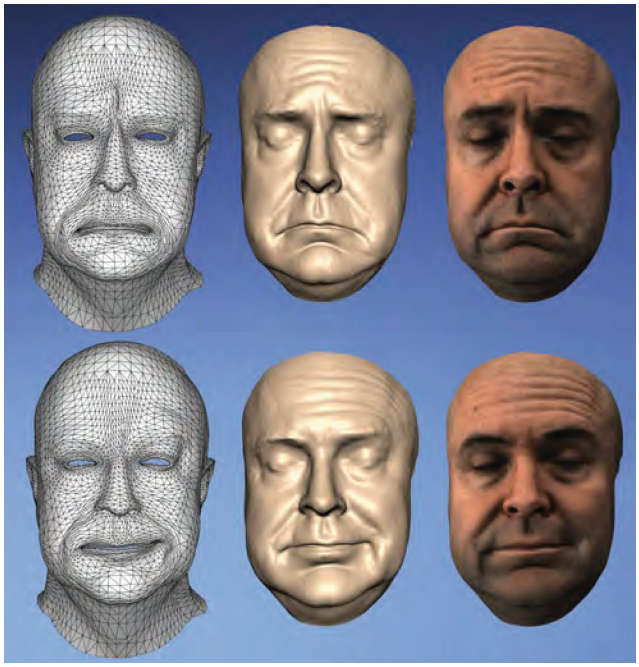

Fig. 10. Our method can enhance a rigged facial performance (left), adding the subtle details of expression particular to an individual's face (shown as a surface and textured).

Fig. 11. Illustrating the shape space matching for one frame of the face rig result from Figure 10. Here we see four of the database poses that are used for interpolation.

A third mode of facial animation that lends itself to our enhancement technique is monocular face tracking using a morphable model [Blanz et al. 2003; Vlasic et al. 2005; Dale et al. 2011]. Here, a low-resolution face model is automatically fitted to a video stream, which can be captured from a handheld camera in outdoor and remote environments (see Figure 12, left). By upsampling this type of animation (Figure 12, right), we demonstrate the ability to achieve studio-quality facial performance capture, even on a moving train. We believe this technology is a large step towards on-set markerless facial motion capture, which can benefit the visual effects industry.

Fig. 12. Result on a morphable model fit to a monocular video sequence [Dale et al. 2011]. From left to right: selected frames from the video, the low-resolution fit model in gray, our enhanced geometry, and our final result rendered with texture.

Finally, we show that our algorithm can enhance facial animations captured using a Kinect depth sensor. Our input is generated from recent technology designed by *faceshift*, based on real-time performance-based facial animation [Weise et al. 2011]. An actor's facial motions drive a low-resolution blendshape model, which we then sparsely sample at 40 locations (see Figure 5) and enhance with our technique. Since the blendshape model is only an approximation of the performance, this result demonstrates the robustness of our approach to handle inaccurately tracked face motion.

The processing time of our algorithm is approximately 30 seconds per frame for our female actress with a mesh resolution of 500K vertices and 55 seconds for the male actor with a mesh resolution of 850K vertices, measured on an i7 desktop machine with 12GB of memory. We use the MOSEK [Andersen and Andersen 2000] library to solve the QP problem (Eq. (2)), and process all four face regions in parallel. Realistic face renders are created using DAZ Studio with the *Elite Human Surface Shader*[2].

Comparison to Ma et al. [2008]. Our work is most similar to the Polynomial Displacement Map (PDM) technique of Ma et al. [2008], however, our method contains some important benefits. The PDM technique is designed for real-time performance on well-tracked input sequences that lie inside the convex hull of a small training set. In that situation, our respective algorithms will produce similar upsampled results. However, in the case that the input shapes are far away from the training set, polynomial extrapolation artifacts can be seen in the method of Ma et al. (see Figure 14). Here we show our algorithm compared to an implementation of the PDM method with the same database on two different inputs, one from Kinect input using faceshift and the other from motion-capture markers. The PDM approach produces unrealistic deformation of the face and amplification of the pore details. One could argue that increasing the size of the training set is a solution, however, the PDM's are determined by an underlying vector field and discontinuities in the vector field cause artifacts in the resulting displacements. The bigger the training set, the harder this vector field is to control. This effect accounts for the discontinuities in the left part of the lip, the left cheek, and the forehead. Finally, in the case of lower-accuracy input sequences like the ones from faceshift (top row of Figure 14), the input deviates again from the training set and results in more artifacts with the PDM approach. Our technique is more general and handles a wider range of scenarios.

[2]www.daz3d.com.

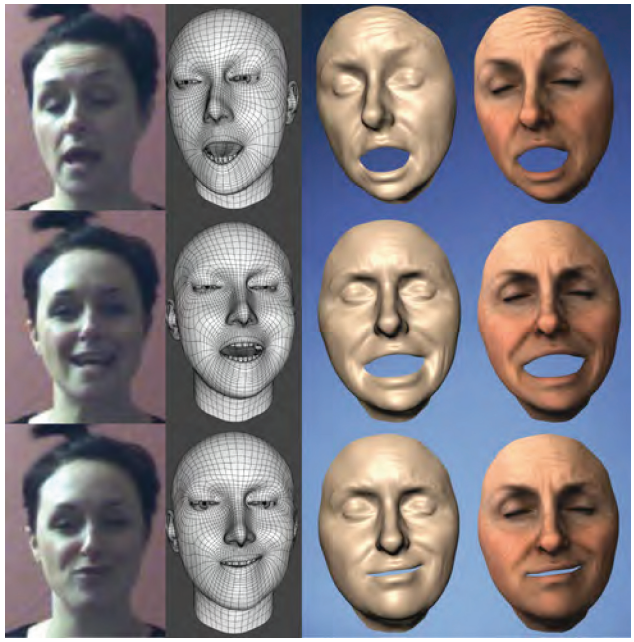

Fig. 13. Enhancement result on Kinect-driven input animations produced by *faceshift* [Weise et al. 2011]. From left to right: reference image from the Kinect, blendshape result of the faceshift software, our enhanced geometry, and our final result rendered with texture.

Limitations and Future Work. One area for future work is to analyze and correct low-frequency errors. Currently, we assume that the low-frequency component of the input animation is correct, however, it could be the case that it does not match the shape and dynamics of the real actor. Furthermore, in this work we use the same actor for the database and input animations to ensure the facial properties are similar. An interesting avenue would be to explore performance transfer, by using input animations from one actor with a database from another. In addition, our method does not currently handle the eye region correctly due to a lack of accurate data in this area. This could be corrected with an improved acquisition system for the database. As a result, we blend closed eyes into all results, which is easily accomplished with our shape space interpolation framework.

8. CONCLUSIONS

We have targeted the gap between low-resolution artistically created facial animations and high-resolution expressive performance capture. On the one hand, art-directed animations are attractive because the animation can easily be tuned for the desired performance. However, they lack the subtle details of deformation and timing that make a real individual's facial performance so expressive and compelling. On the other hand, high-resolution performance capture can acquire the expressive facial details of a performance, but the result can only be played back without further directability. Our method extracts the fine-scale details from a performance capture database that spans the range of expressiveness for a particular individual, and then transfers these details to low-resolution input animations.

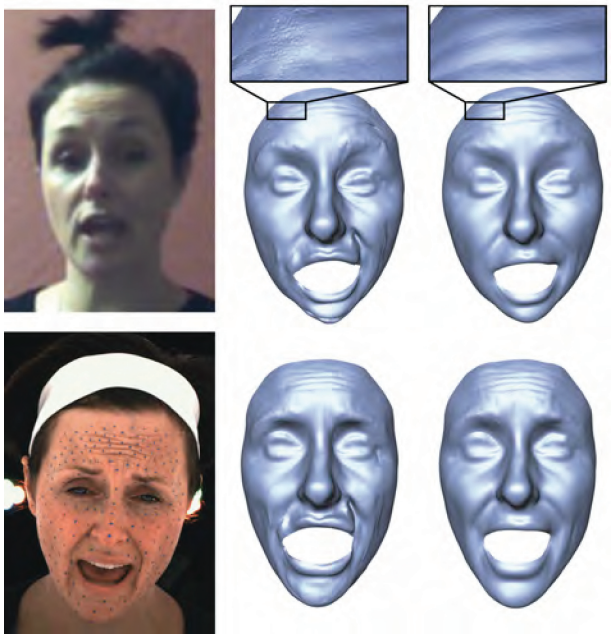

Fig. 14. Comparison to the Polynomial Displacement Map (PDM) technique [Ma et al. 2008] on two different datasets: Kinect input (top row) and motion-capture markers (bottom row). The PDM method (center) exhibits more artifacts around the lips, cheek, forehead and exaggeration of pores, compared to our method (right).

Our system can also improve facial keyframe interpolation so that the dynamics of the real actor are reflected in the enhanced result. We demonstrate our method on four animations created by typical facial animation systems: marker-based motion capture, a hand-animated facial rig, a morphable model fit to monocular video, and a sequence reconstructed with a Kinect depth sensor. We also validate our result against ground-truth data by using a smoothed performance capture animation as input, and provide a direct comparison to current state-of-the-art. With our technique, art-directed animations can now be enhanced to match the expressive quality of performance capture.

ACKNOWLEDGMENTS

We wish to thank our actors, Sean Sutton and Kristen Vermilyea, as well as Maurizio Nitti and Hao Li for cleaning and aligning face scans, Volker Heizle and Kevin Dale for generating input sequences, and Alex Ma and Paul Debevec for helpful insight on the PDM comparison.

REFERENCES

O. Alexander, M. Rogers, W. Lambeth, J.-Y. Chiang, W.-C. Ma, C.-C. Wang, and P. Debevec. 2010. The digital emily project: Achieving a photoreal digital actor. *IEEE Comput. Graph. Appl.* 30, 4, 20–31.

E. D. Andersen and K. D. Andersen. 2000. The mosek interior point optimizer for linear programming: An implementation of the homogeneous algorithm. In *High Performance Optimization*. Kluwer Academic Publishers, 197–232.

I. Baran, D. Vlasic, E. Grinspun, and J. Popovic. 2009. Semantic deformation transfer. *ACM Trans. Graph.* 28, 3, 36:1–36:6.

T. Beeler. B. Bickel, R. Sumner, P. Beardsley, and M. Gross. 2010. High-quality single-shot capture of facial geometry. *ACM Trans. Graph.* 29, 4.

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

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

B. Bickel, M. Lang, M. Botsch, M. A. Otaduy, and M. Gross. 2008. Pose-space animation and transfer of facial details. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 57–66.

V. Basso, C. Poggio, T. Blanz, and T. Vetter. 2003. Reanimating faces in images and video. *Comput. Graph. Forum* 22, 3, 641–650.

G. Borshukov, D. Piponi, O. Larsen J. Lewis, and C. Tempelaar-Lietz. 2003. Universal capture – Image-based facial animation for "the matrix reloaded". In *Proceedings of the ACM SIGGRAPH Sketches and Applications Conference*.

M. Botsch, R. Sumner, M. Pauly, and M. Gross. 2006. Deformation transfer for detail-preserving surface editing. In *Proceedings of the Workshop on Vision, Modeling and Visualization*. 357–364.

D. Bradley, W. Heidrich, T. Popa, and A. Sheffer. 2010. High resolution passive facial performance capture. *ACM Trans. Graph.* 29, 4.

M. Brand. 1999. Voice puppetry. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. 21–28.

C. Bregler, M. Covell, and M. Slaney. 1997. Video rewrite: Driving visual speech with audio. In *Proceedings of the Annual Conference on Computer Graphics (SIGGRAPH'97)*. 353–360.

I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. H. Salesin, J. Seims, R. Szeliski, and K. Toyama. 2000. Performance-driven hand-drawn animation. In *Proceedings of the 1st International Symposium on NonPhotorealistic Animation and Rendering (NPAR'00)*. 101–108.

Y. Cao, P. Faloutsos, E. Kohler, and F. Pighin. 2004. Realtime speech motion synthesis from recorded motions. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 345–353.

J. Chai and J. K. Hodgins. 2007. Constraint-based motion optimization using a statistical dynamic model. *ACM Trans. Graph.* 26, 3, 8:1–8:9.

J.-X. Chai, J. Xiao, and J. Hodgins. 2003. Vision-based control of 3d facial animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 193–206.

E. Chuang and C. Bregler. 2002. Performance driven facial animation using blendshape interpolation. Tech. rep. CS-TR-2002-02, Department of Computer Science, Stanford University.

K. Dale, K. Sunkavalli, M. K. Johnson, D. Vlasic, W. Matusik, and H. Pfister 2011. Video face replacement. In *Proceedings of the SIGGRAPH Asia Conference (SA'11)*. 130:1–130:10.

D. Decarlo and D. Metaxas. 1996. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'96)*. 231–238.

Z. Deng, J. Lewis, and U. Neumann. 2005. Synthesizing speech animation by learning compact speech co-articulation models. In *Proceedings of the Computer Graphics International (CGI'05)*. 19–25.

M. Desbrun, M. Meyer, P. Schroder, and A. H. Barr. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'99)*. ACM Press/Addison-Wesley, 317–324.

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

I. Essa, S. Basu, T. Darrell, and A. Pentland. 1996. Modeling, tracking and interactive animation of faces and heads: Using input from video. In *Proceedings of the Conference on Computer Animation (CA'96)*. 68–79.

T. Ezzat, G. Geiger, and T. Poggio. 2002. Trainable videorealistic speech animation. *ACM Trans. Graph.* 21, 3, 388-398.

W.-W. Feng, B.-U. Kim, and Y. Yu. 2008. Real-time data-driven deformation using kernel canonical correlation analysis. *ACM Trans. Graph.* 27, 3, 91:1–91:9.

A. Golovinskiy, W. Matusik, H. Pfister, S. Rusinkiewicz, and T. Funkhouser. 2006. A statistical model for synthesis of detailed facial geometry. *ACM Trans. Graph.* 25, 3, 1025–1034.

B. K. P. Horn. 1987. Closed-form solution of absolute orientation using unit quaternions. *J. Optical Soc. Amer. A* 4, 4, 629–642.

H. Huang, J. Chai, X. Tong, and H.-T. Wu. 2011a. Leveraging motion capture and 3d scanning for high-fidelity facial performance acquisition. *ACM Trans. Graph.* 30, 4, 74:1–74:10.

H. Huang, L. Zhao, K. Yin, Y. Qi, Y. Yu, and X. Tong. 2011b. Controllable hand deformation from sparse examples with rich details. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM Press, New York, 73–82.

A. Jones, A. Gardner, M. Bolas, I. Mcdowall, and P. Debevec. 2006. Performance geometry capture for spatially varying relighting. In *Proceedings of the 3rd European Conference on Visual Media Production (CVMP'06)*.

S. Kshirsagar and N. M. Thalmann. 2003. Visyllable based speech animation. *Comput. Graph. Forum* 22, 3.

J. Lewis, M. Cordner, and N. Fong. 2000. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'00)*. 165–172.

J. Lewis, J. Mooser, Z. Deng, and U. Neumann. 2005. Reducing blendshape interference by selected motion attenuation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D'05)*.

H. Li, P. Roivainen, and R. Forchheimer. 1993. 3-D motion estimation in model-based facial image coding. *IEEE Trans. Pattern Anal. Mach. Intell.* 15, 6, 545–555.

H. LI, R. W. Sumner, and M. Pauly. 2008. Global correspondence optimization for non-rigid registration of depth scans. *Comput. Graph. Forum* 27, 5.

J. Ma, R. Cole, B. Pellom, W. Ward, and B. Wise. 2004. Accurate automatic visible speech synthesis of arbitrary 3d model based on concatenation of diviseme motion capture data. *Comput. Anim. Virtual Worlds* 15, 1–17.

W.-C. Ma, T. Hawkins, P. Peers, C.-F. Chabert, M. Weiss, and P. Debevec. 2007. Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques (EGSR'07)*. 183–194.

W.-C. Ma, A. Jones, J.-Y. Chiang, T. Hawkins, S. Frederiksen, P. Peers, M. Vukovic, M. Ouhyoung, and P. Debevec. 2008. Facial performance synthesis using deformation-driven polynomial displacement maps. *ACM Trans. Graph.* 27, 5.

X. Ma, B. H. Le, and Z. Deng. 2009. Style learning and transferring for facial animation editing. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'09)*. 123–132.

J.-Y. Noh, and U. Neumann. 2001. Expression cloning. In *Proceedings of the Annual Conference on Computer Graphics (SIGGRAPH'01)*. 277–288.

F. I. Parke. 1974. A parametric model for human faces. Ph.D. thesis, University of Utah.

F. H. Pighin, R. Szeliski, and D. Salesin. 1999. Resynthesizing facial animation through 3d model-based tracking. In *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV'99)*. 143–150.

H. Pyun, Y. Kim, W. Chae, H. Kang, and S. Shin. 2003. An example-based approach for facial expression cloning. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 167–176.

Y. Seol, J. Seo, P. H. Kim, J. P. Lewis, and J. Noh. 2011. Artist friendly facial animation retargeting. *ACM Trans. Graph.* 30, 6.

E. Sifakis, I. Neverov, and R. Fedkiw. 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph.* 24, 3, 417–425.

O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rossl, and H.-P. Seidel. 2004. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP'04)*. ACM Press, New York, 179–188.

B. Sumner and J. Popovic. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph.* 23, 3, 399–405.

K. Takayama, R. Schmidt, K. Singh, T. Igarashi, T. Boubekeur, and O. Sorkine. 2011. Geobrush: Interactive mesh geometry cloning. *Comput. Graph. Forum* 30, 2, 613–622.

J. R. Tena, F. D. L. Torre, and I. Matthews. 2011. Interactive region-based linear 3d face models. *ACM Trans. Graph.* 30, 4.

D. Terzopoulus and K. Waters. 1993. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Trans. Pattern Anal. Mach. Intell.* 14, 569–579.

K. Venkataraman, S. Lodha, and R. Raghavan. 2005. A kinematic-variational model for animating skin with wrinkles. *Comput. Graph.* 29, 5, 756–770.

D. Vlasic, M. Brand, H. Pfister, and J. Popovic. 2005. Face transfer with multilinear models. *ACM Trans. Graph.* 24, 3, 426–433.

Y. Wang, X. Huang, C.-S. Lee, S. Zhang, Z. Li, D. Samaras, D. Metaxas, A. Elgammal, and P. Huang. 2004. High resolution acquisition, learning and transfer of dynamic 3-d facial expressions. *Comput. Graph. Forum* 23, 3, 677–686.

K. Waters. 1987. A muscle model for animating three-dimensional facial expression. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'87)*. 17–24.

T. Weise, S. Bouaziz, H. Li, and M. Pauly. 2011. Realtime performance-based facial animation. *ACM Trans. Graph.* 30, 4.

A. Wenger, A. Gardner, C. Tchou, J. Unger, T. Hawkins, and P. Debevec. 2005. Performance relighting and reflectance transformation with time-multiplexed illumination. *ACM Trans. Graph.* 24, 3, 756–764.

T. Weyrich, W. Matusik, H. Pfister, B. Bickel, C. Donner, C. Tu, J. Mcandless, J. Lee, A. Ngan, H. W. Jensen, and M. Gross. 2006. Analysis of human faces using a measurement-based skin reflectance model. *ACM Trans. Graph.* 25, 3, 1013–1024.

L. Williams. 1990. Performance-driven facial animation. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'90)*. 235–242.

C. Wilson, A. Ghosh, P. Peers, J.-Y. Chiang, J. Busch, and P. Debevec 2010. Temporal upsampling of performance geometry using photometric alignment. *Trans. Graph.* 29, 2.

Y. Wu, P. Kalra, and N. Magnenat-Thalmann. 1996. Simulation of static and dynamic wrinkles of skin. In *Proceedings of the Conference on Computer Animation (CA'96)*. 90–97.

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

S. Zhang and P. Huang. 2006. High-resolution, real-time three-dimensional shape measurement. *Optical Engin.* 45, 12.

Y. Zhang and T. Sim. 2005. Realistic and efficient wrinkle simulation using an anatomy-based face model with adaptive refinement. In *Proceedings of the Computer Graphics International (CGI'05)*. 3–10.

Received November 2012; revised July 2013; accepted August 2013