High Resolution Passive Facial Performance Capture

Derek Bradley[1] Wolfgang Heidrich[1] Tiberiu Popa[1,2] Alla Sheffer[1]
1) University of British Columbia 2) ETH Zürich

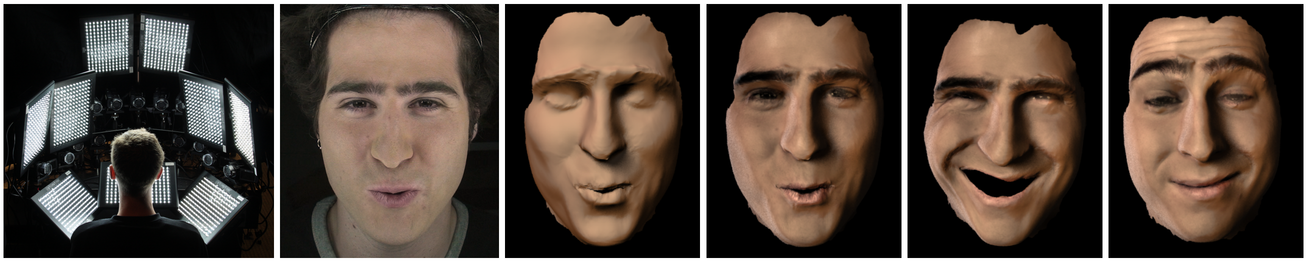

Figure 1: *High resolution passive facial performance capture. From left to right: Acquisition setup; one reference frame; the reconstructed geometry (1 million polygons); final textured result; and two different frames.*

Abstract

We introduce a purely passive facial capture approach that uses only an array of video cameras, but requires no template facial geometry, no special makeup or markers, and no active lighting. We obtain initial geometry using multi-view stereo, and then use a novel approach for automatically tracking texture detail across the frames. As a result, we obtain a high-resolution sequence of compatibly triangulated and parameterized meshes. The resulting sequence can be rendered with dynamically captured textures, while also consistently applying texture changes such as virtual makeup.

CR Categories: I.3.3 [COMPUTER GRAPHICS]: Picture/Image Generation—Digitizing and scanning; I.3.5 [COMPUTER GRAPHICS]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems.

Keywords: face reconstruction, performance capture, markerless motion capture

1 Introduction

Facial performance capture is evolving into a major tool for creating realistic animations in both movie and game industries. A practical and versatile facial performance capture system should fulfill a number of requirements. First, it should generate sequences of detailed meshes with dynamic texture in order to capture both geometric deformations of the face, as well as the corresponding changes in skin appearance due to sweating or changes in blood circulation. Second, the triangulation of the meshes and their mapping between frames should be compatible over time, such that both the geometry and the texture can be edited by an artist (e.g. by applying virtual makeup), and these changes can be propagated over time. Finally, the capture process itself should be simple and automatic, and should not require separate geometry scans or excessively expensive hardware.

Reconstructing a human face is challenging, since most faces do not have sufficient medium-scale texture to establish dense correspondences between different viewpoints. For this reason, commonly-used methods typically involve either structured lighting [Zhang et al. 2004; Wang et al. 2004], special makeup [Furukawa and Ponce 2009], or markers [Bickel et al. 2007; Ma et al. 2008] to track the geometry. This makes it difficult to simultaneously capture both geometry and texture, thus requiring either inpainting of markers or sacrificing temporal resolution by staggering structured light with uniform light. Additionally, these active methods can be uncomfortable for the actors, affecting their performance. Many of these techniques also require an initial laser scan of the face, while the others suffer from low-resolution reconstructions.

In this paper, we present a fully passive method for facial performance capture that satisfies the criteria outlined above. By using only a camera array and uniform illumination, our setup is less intrusive to the actors. We require no markers, no face paint, no fluorescent makeup, no structured light, and no laser-scanned model, yet are still able to reconstruct high resolution time-varying meshes at 30 frames per second. Our passive setup also allows us to reconstruct high-resolution, time-varying texture maps that can capture potential changes in skin appearance due to, for example, blushing or sweating of the actor. Our reconstruction is made possible by a number of contributions:

1. A novel high-resolution acquisition setup that allows us to use pores, blemishes and hair follicles as trackable features.
2. A high-quality stereo reconstruction algorithm, extending a current state-of-the-art technique [Bradley et al. 2008a] to include new disparity constraints that work with our setup.
3. Most significantly, an automatic surface tracking method based on optical flow. This method supports automatic drift correction and edge-based mouth tracking, which together yield realistic, time-varying, textured facial models.

Our capture system consists of the three main components shown in Figure 2:

- **Acquisition setup** - Our setup consists of 14 high definition video cameras, arranged in seven binocular stereo pairs. Each pair is zoomed-in to capture a small patch of the face surface in high detail under bright ambient illumination (Section 3).

ACM Reference Format
Bradley, D., Heidrich, W., Popa, T., Sheffer, A. 2010. High Resolution Passive Facial Performance Capture. *ACM Trans. Graph. 29*, 4, Article 41 (July 2010), 10 pages. DOI = 10.1145/1778765.1778778 http://doi.acm.org/10.1145/1778765.1778778.

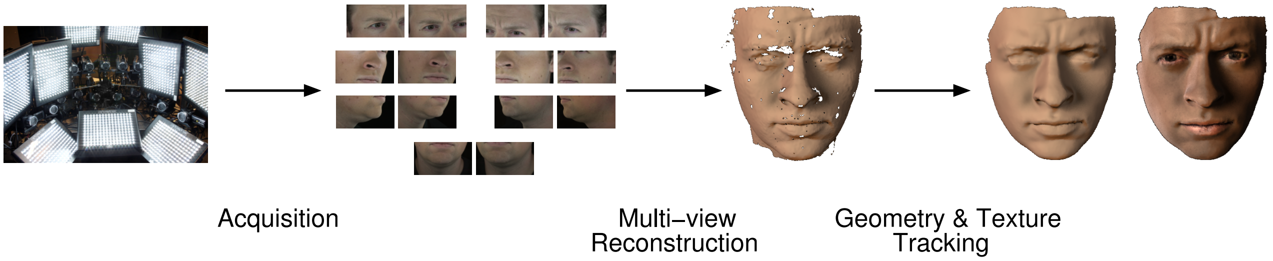

Figure 2: *Overview of our algorithm.*

- **Multi-view reconstruction** - We use an iterative binocular stereo method to reconstruct each of the seven surface patches independently, and then combine them into a single high-resolution mesh. The zoomed-in cameras allow us to use skin pores, hair follicles and blemishes as surface texture to guide the stereo algorithm, producing meshes with roughly 1 million polygons (Section 4).
- **Geometry and texture tracking** - In order to consistently track geometry and texture over time, we choose a single reference mesh from the sequence, and compute a mapping between it and every other frame by sequentially using optical flow. The observed pores and other surface details not only serve to provide accurate per-frame reconstructions, but also allow us to compute cross-frame flow. Drift caused by inevitable optical flow error is detected in the per-frame texture maps and corrected in the geometry. In order to account for the high-speed motion generated by talking, the mapping is guided by an edge-based mouth-tracking process (Section 5).

2 Related Work

Previous techniques for facial performance capture can largely be divided into the following categories: methods that use markers or special face paint to guide the reconstruction, structured light approaches, and finally methods that fit parametric face models to observed video sequences.

Markers and Face Paint. Traditional marker-based face capture dates back to Williams [1990], and consists of covering the face with a large number of black dots or fluorescent colors [Guenter et al. 1998; Lin and Ouhyoung 2005] in order to track the geometry. Bickel et al. [2007] augment the traditional methods to incorporate multiple scales of geometry and motion using markers, face paint, and an initial scan. Most recently, Furukawa and Ponce [2009] present a face capture technique that deforms a laser-scanned model to match a highly-painted face while regularizing non-rigid tangential deformations.

Unfortunately, these techniques have trouble reconstructing accurate per-frame face textures, and require expensive hardware to perform initial scans.

Structured Light. Another approach to facial performance capture is to combine cameras with at least one projector that casts a structured light pattern onto the face in order to provide dense surface texture. Zhang et al. [2004] use space-time stereo with structured light to reconstruct temporally smooth depth maps of a face. They then fit a deformable template model at each time step using optical flow. Wang et al. [2004] project phase-shifted color-fringe patterns onto the face and acquire the 3D shape in real-time. They establish correspondences between frames using a multi-resolution model fitting approach. However, the resulting meshes have insufficient resolution (at most 16K vertices) for capturing fine-scale facial details such as wrinkles. Ma et al. [2008] achieve high-resolution reconstructions by interleaving structured light with spherical gradient photometric stereo [Ma et al. 2007] using the USC Light Stage. New facial performances are then synthesized using a marker-based, data-driven approach. However, this method requires expensive hardware.

Structured light approaches are unattractive because they can be distracting to the actor, and they suffer from the inability to reconstruct face textures without sacrificing temporal resolution by interleaving ambient illumination with the structured light.

Parametric Models from Video. The goal of this category of methods is to determine the parameters of a deformable face model from observing a video sequence without markers or structured light [Li et al. 1993; Essa et al. 1996; DeCarlo and Metaxas 1996; Pighin et al. 1999]. However, the resulting face reconstructions tend to be very low resolution, lacking any person-specific details.

In a related work, Blanz et al. [2003] re-animate faces in video by parameterizing a database of different laser-scanned faces and expressions. They can then estimate the 3D shape and pose of new faces in video images with a parameter fitting algorithm. But like the other parametric approaches, the final models tend to lack facial details.

Commercial Systems and Other Projects. A number of commercial systems have been developed for facial performance capture. Vicon's marker-based system[1] and Mova's fluorescent makeup CONTOUR Reality Capture[2] are two prominent examples, while Dimension Imaging 3D is among the first to use markerless facial reconstruction in industry[3]. Recently, Alexander et al. [2009] created a photoreal facial modeling and animation system in the Digital Emily Project. Finally, Borshukov et al. [2003] recreate actors for *The Matrix Reloaded* using their Universal Capture system. These systems both start with laser-scanned models. The approach of Borshukov is most similar to ours. Optical flow and camera triangulation advances a face model over time, and a time-varying texture map is computed from multiple videos. Unlike our automatic approach, however, optical flow errors and drift are corrected using tedious manual geometry reshaping.

To our knowledge, ours is the first face capture method with fully-automatic temporal reconstruction, rivaling current state-of-the-art techniques, but without the need for markers, face paint, expensive hardware or structured light. It is worth noting that concurrent to our work, Beeler et al. [2010] present a similar technique for passive face reconstruction that captures pore-scale geometry of static faces. Our automatic surface tracking method for generating temporally compatible animations could complement their approach.

[1] Vicon MX - www.vicon.com

[2] CONTOUR Reality Capture - www.mova.com

[3] www.di3d.com

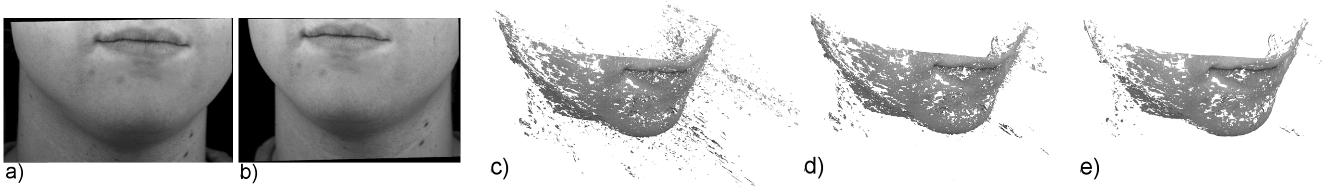

Figure 3: *Iteratively constrained multi-view stereo removes outliers by iteratively tightening depth constraints.*

3 Acquisition Setup

Our acquisition setup consists of 14 high definition Sony HDR-SR7 cameras arranged in an array in front of the actor (see Figure 4). The cameras are geometrically calibrated [Bradley and Heidrich 2010] and optically synchronized [Bradley et al. 2009]. We use nine LED light fixtures, each with 192 LEDs, to provide both bright, uniform illumination, as well as the camera synchronization.

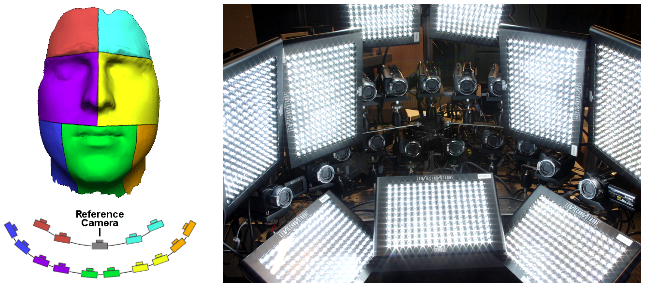

Figure 4: *Schematic view and photo of our acquisition setup.*

As we mentioned previously, reconstructing a human face is challenging because most faces do not have sufficient medium-scale surface texture. This is the main reason why previous work relies on hand-placed markers or structured light patterns. However, this assumption does not hold if we increase the resolution of the acquisition system and capture images of very small face details such as pores, freckles and wrinkles.

To this end, we arrange the cameras in seven stereo pairs and maximize the optical zoom level in order to observe fine-scale surface details, providing a natural surface texture for reconstruction. Figure 4 shows an illustration of the setup and the seven face regions observed by the stereo pairs, along with a photo of the actual setup. Note that there is significant overlap between the face regions, even though it is not shown in the illustration. Figure 5 shows a video frame captured from one camera, demonstrating the high resolution surface details. We also have an additional *reference* camera that is not zoomed-in and not used for processing. This camera provides a normal video for comparison with our results.

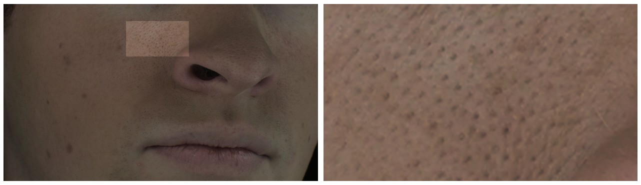

Figure 5: *One HD video frame showing that pores can be used as natural surface texture for face reconstruction.*

Preparing an actor for capture requires very little work, since we do not place markers on the face or require an initial scan. We do use off-the-shelf foundation makeup to reduce specularity in the case of oily skin, however this is common practice for preparing actors for any performance.

4 Multi-View Reconstruction

To reconstruct a mesh for each face frame, we perform binocular stereo for each of the seven facial regions, resulting in seven overlapping depth images. The natural surface texture obtained from our capture setup provides sufficient detail for stereo reconstruction. The seven depth images are then merged into a single triangle mesh using the multi-view reconstruction system of Bradley et al. [2008a]. This system was successfully used for garment capture [Bradley et al. 2008b], which has similar requirements on accuracy and efficiency for capturing deformable geometry.

However the method of Bradley et al. [2008a] is designed for 360° reconstruction, where the visual hull of the object can be precomputed and used to reduce outliers in the binocular reconstructions. Our camera setup prevents us from applying the technique exactly as described, because we observe only patches of the face and do not have full 360° coverage. Thus we cannot compute a visual hull. Without the visual hull constraint, the resulting depth images can contain many outliers (Figure 3, c). We resolve this problem by adopting an *iteratively constrained* binocular reconstruction approach, designed to iteratively remove outliers in the reconstruction. We apply this method to each of the seven camera pairs.

Iteratively Constrained Binocular Reconstruction. In the first pass of binocular reconstruction, we enforce a loose depth constraint that corresponds to the maximum reconstruction volume for the face patch (measured by hand). The resulting depth image has many valid samples but also contains outliers. Figure 3 (a) and (b) shows an example stereo pair, and the depth outliers can be seen in the corresponding point cloud in Figure 3 (c). In order to reduce outliers, we tighten the initial depth constraints for the next iteration. To this end, we perform Gaussian smoothing on the current depth image using a large kernel size, creating an over-smooth approximation of the surface with fewer outliers. This smooth depth image is used to compute per-pixel constraints for a second pass of binocular stereo. During the second pass, we process only the pixels whose depth in the first reconstruction violates the new constraints. If the depth is within an acceptable distance from the smooth surface then we do not re-process the pixel. By tightening the depth constraints, the depth image computed in the second pass has fewer outliers (Figure 3, d). We repeat the smoothing process on the new depth image to establish new constraints for the next pass, iterating between stereo matching and constraint tightening until convergence. In practice, we found that three iterations was sufficient for all reconstructions (Figure 3, e).

In this approach, we assume that the true surface lies within a small distance of the over-smoothed one in each iteration. This holds true if the surface does not contain very high curvature or sharp features, which is the case for faces. We also assume that outliers have small local support, so that they are removed by the smoothing step.

Pair Merging. After applying iterative binocular reconstruction for each camera pair, we obtain seven corresponding point cloud reconstructions. The point clouds are then merged into a single dense point cloud. On average, we reconstruct approximately 8-10 million points per face. These points are downsampled, filtered and

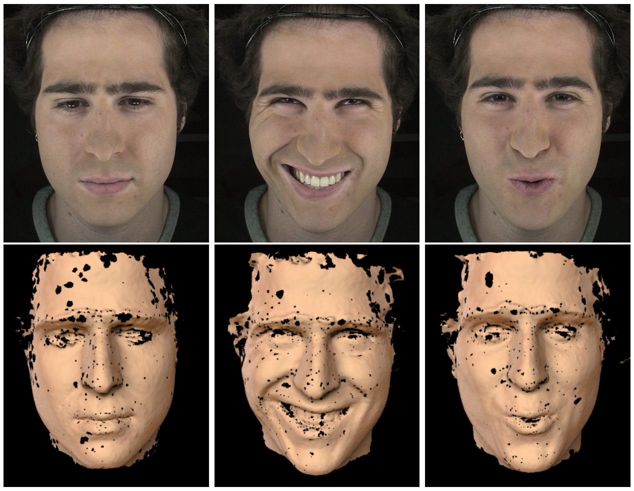

Figure 6: *Multi-view reconstructions for three different frames.*

triangulated using the system of Bradley et al. [2008a] without further modification. Our final meshes have approximately 500K vertices (1 million triangles). Each frame t of a capture sequence can be reconstructed independently, so we perform the reconstructions in parallel. In the following we refer to these initial meshes as G^t. The reconstruction of a selection of frames is shown in Figure 6.

5 Geometry and Texture Tracking

Given the per-frame reconstructions, the next step is to reconstruct the motion of the face by tracking the geometry and texture over time. We explicitly compute a sequence of compatible meshes without holes, which allows artists to later edit both the geometry and the texture, and to propagate these modifications consistently over time. Given the initial per-frame reconstructions G^t, we would like to generate a set of compatible meshes M^t that have the same connectivity as well as explicit vertex correspondence. That is to say, we desire one mesh that deforms over time. In order to create high-quality renderings, we also require per-frame texture maps T^t that capture appearance changes such as wrinkles and sweating of the actor.

We propose an optical flow based approach for motion reconstruction, as illustrated in Figure 7. The basic method works as follows: starting with a single reference mesh M^0, generated by manually cleaning up the first frame G^0 (Section 5.1), we compute dense optical flow on the video images and use it in combination with the initial geometric reconstructions G^t to automatically propagate M^0 through time (Section 5.2). At each time step, we compute a high-quality 2D face texture T^t from the video images (Section 5.3).

In theory this basic method can reconstruct the face motion and produce a temporally consistent animation. However, the practical limitations of optical flow can pose problems. For instance, optical flow is prone to errors during rapid deformations such as lip movement during speech, and can be inaccurate for a variety of other reasons including occlusions, insufficient image details, and appearance changes such as the formation of wrinkles. Furthermore, since the basic method processes the frames sequentially, even the smallest error will accumulate over time, causing the geometry to drift. Temporal drift is unacceptable, as it will destroy the consistent mapping between frames that we aim to reconstruct. We overcome these issues with two improvements to the basic method, which aim to stabilize the animation through explicit mouth tracking and texture-based drift correction (Section 5.4). As a final step, we perform smoothing to remove unwanted noise in the animation (Section 5.5).

5.1 Reference Mesh

We start each performance with a neutral facial expression, from which we create our reference mesh. The first reconstruction, G^0, (shown in Figure 6 left) is manually edited to remove any outliers caused by hair, and then a 2D parameterization of the geometry is computed. We use LSCM [Lévy et al. 2002] to generate the parameterization because it successfully deals with the holes in our initial reconstruction. The holes are filled in 2D by creating small Delaunay triangulations [Shewchuk 1996], and the new 3D geometry is created as a membrane surface, C^1 continuous with the surrounding geometry, using the Laplacian formulation of Bradley et al. [2008b]. Finally, a slit is cut in the mesh for the mouth. The result is the first mesh M^0 of the final sequence. The 2D parameterization computed here will become the domain for the texture map. Note that this parameterization remains constant for the entire sequence. That is to say, as a vertex v_i of the mesh moves over time, its texture coordinates never change.

5.2 Frame Propagation (Basic Method)

We compute optical flow [Bouguet 1999] over the whole sequence for each of the 14 video cameras individually. Using this flow and the initial reconstructions G^t, we can now propagate M^0 forward in time to produce our output sequence. The process is illustrated in Figure 8, and it proceeds as follows. For each vertex v_i^{t-1} of M^{t-1} we project the vertex onto each camera c in which it is visible (i.e. inside the field of view of and not occluded). Let $p_{i,c}$ be this projected pixel. We then look up the 2D flow vector that corresponds to $p_{i,c}$ and add the flow to get a new pixel location $p'_{i,c}$. Back-projecting from $p'_{i,c}$ onto G^t gives us a guess for the new vertex location, which we call $\bar{v}_{i,c}^t$. The illustration in Figure 8 has exaggerated inter-frame motion for better visualization.

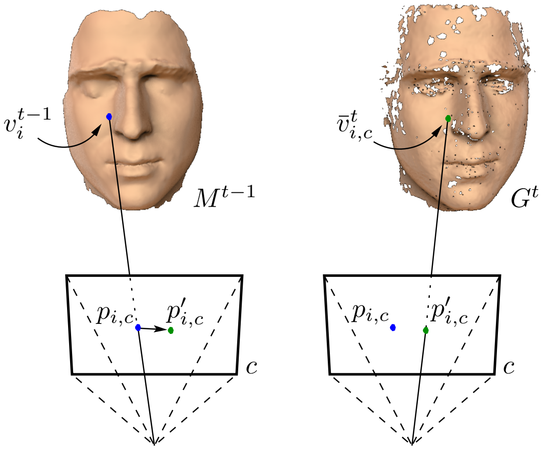

Figure 8: *Computing vertex positions for the next frame, using per-camera optical flow (with exaggerated motion for visualization).*

We require at least two cameras to agree on the new vertex location. We say that cameras c_1 and c_2 agree if

$$\| \bar{v}_{i,c_1}^t - \bar{v}_{i,c_2}^t \| < 1mm. \tag{1}$$

The computed vertex location $\bar{v}_i^t$ is then a weighted average of the n per-camera guesses that agree:

$$\bar{v}_i^t = \sum_{c=1}^{n} w_{i,c}^t \cdot \bar{v}_{i,c}^t, \tag{2}$$

where $w_{i,c}^t$ is the dot product between the surface normal at $\bar{v}_{i,c}^t$ and the vector from there to c. The difference between the new vertex location and its previous location can be considered a 3D flow

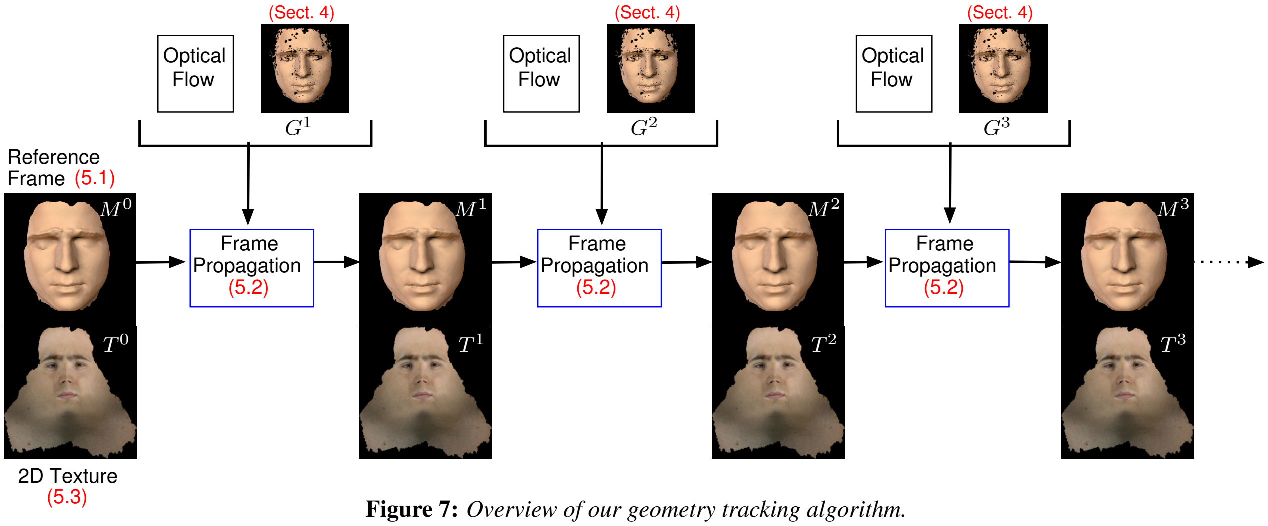

Figure 7: *Overview of our geometry tracking algorithm.*

vector, which we denote δ_i^t. If not enough cameras agree on a new vertex location, for example if a vertex falls on a hole in the next frame, then the 3D flow from neighboring vertices is interpolated. We assume that the motion of the face is spatially smooth, so neighboring vertices have similar 3D flow. The interpolation is achieved by solving a simple least-squares Laplacian system on the surface for all vertices that were not updated (all updated vertices remain fixed):

$$\min \| \Delta\delta_i^t \|^2 . \tag{3}$$

Finally, we apply regularization to the mesh in order to avoid possible triangle-flips and remove any unwanted artifacts that may have been present in the initial reconstruction. Following the regularization method of de Aguiar et al. [2008], we again solve a least-squares Laplacian system using cotangent weights and the current positional constraints $\bar{v}_i^t$. Thus, we generate the final mesh M^t by minimizing

$$\arg\min_{v^t}\{\| v_i^t - \bar{v}_i^t \|^2 + \alpha \| Lv^t - Lv^0 \|^2\}, \tag{4}$$

where L is the cotangent Laplacian matrix. The parameter α controls the amount of regularization, and is set to 100 for all reconstructions.

5.3 Computing 2D Texture

At each time step, in addition to reconstructing geometry, we also compute a high-resolution 2D texture T^t for rendering. Since we do not use markers or face paint, our texture images are rich in detail, containing per-frame appearance changes due to, for example, blushing or sweating of the actor. As pointed out by Borshukov et al. [2003], these textural variations are very important for creating believable facial renderings.

All of our 14 HD cameras are used to compute a single texture that covers the entire surface. This allows us to create very high-resolution textures, in the order of 8-10 megapixels, for extreme zooms (see Figure 17). For other purposes, lower resolution textures may be sufficient (e.g. 900×900 as used in most of our examples). The domain of the texture image is given by the 2D parameterization of the mesh (Section 5.1). Every vertex of the mesh has unique 2D coordinates in the parameter domain, yielding a one-to-one mapping between 2D and 3D mesh triangles.

To compute the texture for frame t, we start by projecting each triangle of M^t onto the camera that observes it best, as determined by the dot product between the triangle normal and the camera direction. The camera pixels corresponding to the projection are then copied to the corresponding 2D triangle in the texture domain. Figure 9 shows the computation of a face texture. We visualize the contribution from each camera as a grayscale *patch* image in Figure 9 (middle-left). Figure 9 (middle-right) shows the initial texture result after copying the pixels from the camera images. Since the cameras are not radiometrically calibrated, different texture patches can have drastically different skin tones. To compute the final texture, shown in Figure 9 (far right), we apply Poisson image editing [Pérez et al. 2003] similar to Mohammed et al. [2009]. We start with the largest patch and iteratively add adjacent patches until the texture image is complete. For each new patch we compute x- and y-gradients inside the patch and solve a Poisson equation to find a new patch that matches the gradients as closely as possible, while also obeying the boundary conditions set by other completed patches. Finally, in order to have temporally consistent textures, we use the previous texture T^{t-1} as per-pixel soft constraints when solving the Poisson equation.

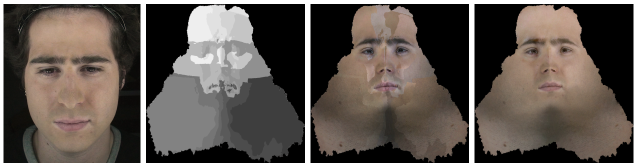

Figure 9: *2D texture generation. From left to right: reference image, camera contribution image, initial texture, final texture.*

Two additional textures for the same sequence are shown in Figure 10. Notice how the texture domain remains fixed even though the 3D face undergoes substantial deformations, including opening of the mouth. The only differences in the textures is changes in skin appearance caused by wrinkles, blushing or sweating of the actor.

5.4 Tracking Enhancements

In general, the basic optical flow-based tracking technique described so far produces realistic animations of face deformation. For most of the face, optical flow vectors are both dense and accurate, since our capture setup provides natural high-resolution surface features which easily guide the flow computation. However,

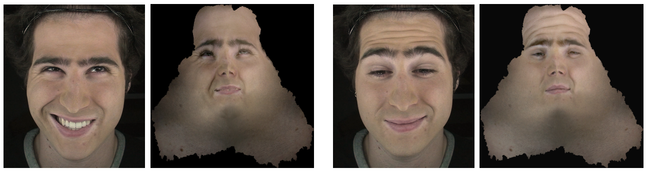

Figure 10: *Texture results for two different frames, including reference images.*

optical flow can fail during very fast motion such as rapid mouth deformations, and minor inaccuracies can accumulate over time leading to temporal drift. We resolve these problems automatically by enhancing the basic technique with an explicit mouth tracking algorithm, along with a method for detecting and correcting temporal drift.

5.4.1 Mouth Tracking

To perform automatic mouth tracking, we introduce positional constraints for a sparse set of points around the mouth at each time step. The positional constraints are computed in image-space, and thus do not map directly to mesh vertices. Instead, they are incorporated as barycentric constraints on mesh triangles.

The constraint points are determined by tracking the mouth in a single camera (either one of the two *green* cameras in Figure 4). We perform edge detection [Canny 1986] on each frame within a user-specified region-of-interest (ROI). The region should contain the mouth throughout the sequence but avoid other edges caused by surrounding wrinkles or the silhouette of the face. If the sequence contains too much global face motion to contain the mouth in a single ROI then mouth tracking can be performed in temporal segments with different ROIs. For each frame we perform a simple analysis of the detected edges to locate the mouth. Figure 11 shows a few different frames of mouth tracking. Detected edges are shown in white, and the ROI is indicated by the blue rectangle. If we consider the image as a set of rows and columns, we start by choosing the two corners of the mouth (shown as red points) as the minimum and maximum columns that contain an edge pixel. We then detect the top and bottom lips by uniformly sampling a sparse number of columns between the mouth corners, and selecting the minimum and maximum rows at each column that contain edge pixels (shown as green points). Empirically we found that 30 sample columns were sufficient. These 62 pixels then become the mouth constraints for this frame. We process each frame in the same manner, yielding an explicit temporal correspondence between each of the individual constraints. Since edge detection is notoriously unstable, we smooth the constraints both spatially and temporally to remove outliers. Although our mouth tracking technique is rather simple, we found the results to be quite robust, as we show in the bottom row of Figure 11.

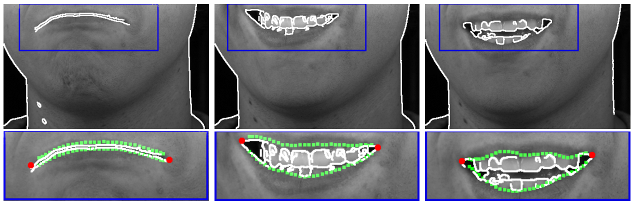

Figure 11: *Mouth tracking through edge detection. The green and red points become constraints in geometry tracking.*

We back-project the constrained pixels into the reference frame M^0 to determine the set of constraint mesh triangles and the corresponding barycentric coordinates for each of the mouth constraint points. We encode the barycentric coordinates in a sparse matrix B which has similar structure to the Laplacian matrix, except that it contains rows only for vertices that are adjacent to a constraint triangle. Let P^0 be the set of 3D constraint points determined from the back-projection, then

$$P^0 = Bv^0. \tag{5}$$

Throughout the sequence B remains fixed. The per-frame mouth constraints are used to compute 3D constraint points P^t by back-projecting onto the initial reconstructions G^t. The mouth constraints guide the regularization from Section 5.2, replacing Equation 4 with

$$\arg\min_{v^t} \{ \| v_i^t - \bar{v}_i^t \|^2 + \alpha \| Lv^t - Lv^0 \|^2 + \beta \| Bv^t - P^t \|^2 \}, \tag{6}$$

where β controls how much we constrain the mouth. We use a high weight, $\beta = 10^4$, since the mouth can deform quite rapidly, causing large errors in the basic optical flow approach. Our mouth tracking procedure alleviates these errors, and thus is an essential part of our method for generating realistic facial animations.

5.4.2 Drift Correction

It is well-known that optical-flow based tracking methods suffer from accumulation of error, known as drift [DeCarlo and Metaxas 2000; Borshukov et al. 2003]. DeCarlo and Metaxas [1996] solve this problem by combining optical flow with edge information, and Borshukov et al. [2003] rely on manual intervention. A key feature of our method is that we are able to detect and correct drift in the 3D animation automatically, using the texture domain of the faces.

Drift typically occurs because optical flow is computed between successive video frames only. If it were possible to accurately compute flow between the first video image and every other frame, there would be no accumulation of error. Unfortunately, temporally distant video images in a capture sequence are usually too dissimilar to consider this option. In our case, however, the texture domain of the mesh remains constant over time, which means that the computed per-frame texture images are all very similar. Any temporal drift in the 3D geometry appears as a small 2D shift in the texture images, which can easily be detected, again by optical flow.

To incorporate drift correction, we employ a simple modification to the basic tracking method described in Section 5.2. After computing the geometry M^t and texture T^t for a given frame, we compute optical flow between the textures T^0 and T^t. This flow (if any is detected) is then used to update M^t on a per-vertex basis using the direct mapping between the geometry and the texture. Any shift in texture space becomes a 3D shift along the mesh surface. After updating the vertices to account for drift we apply regularization again (Equation 6), to avoid possible triangle flips.

The only problem that remains is that, if significant appearance changes such as wrinkles have occurred in the current frame, optical flow between T^0 and T^t can fail, resulting in large flow errors. However, since we expect drift to appear gradually, the flow between T^0 and T^t should never be more than a few pixels. Larger flow vectors are discarded as outliers. Still, face regions that contain these appearance changes may incur drift, as wrinkles can be present for a significant number of frames. In these regions, we detect drift more locally by computing the flow between T^{t-k} and T^t, and updating the geometry accordingly. We choose k to be small, so that both frames $(t-k)$ and t contain similar appearance, such as the same wrinkles, allowing flow to be computed accurately. On the other hand, k must be large enough so that drift can accumulate and be detected. In all reconstructions, we found that $k = 5$ was an appropriate trade-off. The decision to switch from global to local drift correction is made automatically, on a per pixel basis, when there is no valid optical flow between T^0 and T^t. By performing local drift correction we are, in effect, only slowing down the drift accumulation rather than removing it. However, this approach does stabilize the animation until the wrinkles disappear, at which time normal drift correction is automatically resumed.

5.5 Post-Processing

As a final step, we post-process the sequence to provide a smooth, realistic facial animation.

Saliency-Based Smoothing. As with any stereo reconstruction method, spatial noise can appear in the resulting geometry due to, for example, slight inaccuracies in camera calibration (see Figure 12-bottom left). We wish to smooth the face meshes to remove this noise, but avoid removing the spatial features and wrinkles that define the face. To accomplish this, we introduce saliency-based smoothing, a technique for smoothing less-salient regions of the face while preserving more-salient features. Saliency is computed in the texture image T^t through a simple analysis of local histograms. Kadir and Brady [2001] remark that areas of an image with high saliency tend to have flatter distributions in the local histogram of intensities. Following this principle, we mark a pixel in T^t as non-salient if its local histogram contains a single strong peak. All other pixels are salient. We compute histograms in local 15x15 windows, quantized to 8 intensity bins and use simple thresholding to determine if a peak exists. Figure 12 (top row) shows one of the texture images and the computed saliency mask, where salient pixels are white. The saliency mask is then used to constrain salient vertex positions in a Laplacian smoothing step. The result is shown in the bottom row of Figure 12. Notice that the eyebrows, mouth and cheek deformation were not affected by the smoothing.

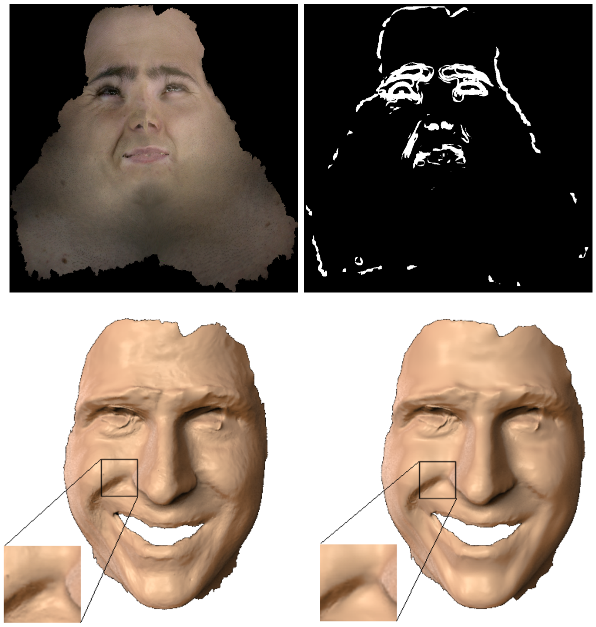

Figure 12: *Saliency-based spatial smoothing. Top: an input texture and the saliency map. Bottom: before and after smoothing.*

Relaxing the Eye Geometry. As we saw in the initial reconstructions (Figure 6) and again in Figure 12, the eye regions are not reconstructed well. This is because eyes are too specular, and surrounding eye-lashes are thin hairs that only add noise to the surface. We alleviate this problem by performing localized smoothing in the eye regions, as shown in Figure 13. These eye regions are marked manually in the first frame and are then propagated automatically for the rest of the sequence.

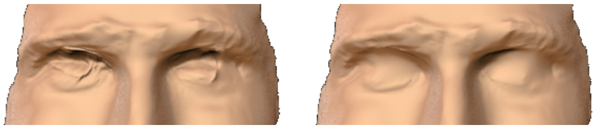

Figure 13: *Local smoothing to improve the eye regions.*

Temporal Smoothing. We end with a single pass of Gaussian smoothing in the temporal domain to prevent temporal flicker. In practice we found that temporal noise was minimal, so we use a small smoothing radius of only three frames.

After all post-processing, 2D textures are re-computed since the face geometry has changed.

6 Results

We now show results for a number of facial performances given by different people. All of our results were rendered with Renderman, using the MakeHuman skin shader in Pixie[4]. We encourage the reader to also view the accompanying video, which further demonstrates our results.

Figure 18 contains six frames from the first sequence, rendered in a number of different ways to highlight the results. The top row is the reference footage. In the second row we show the geometry without a texture map. Here we can see the exact geometric details that form each facial expression. The third row is rendered using a static checkerboard texture. This visualization shows how the skin stretches and compresses during deformation, for example when raising the eyebrows in the second image from the left. The stability of the checker pattern over time also indicates that we have achieved a drift-free reconstruction. The fourth row shows our high-quality rendering using the captured per-frame textures. Finally, the last row demonstrates how virtual makeup can be applied to the sequence. Here, an artist would edit the makeup in the first frame and the edits would be propagated to the rest of the sequence automatically. Once a facial performance has been captured, we can also render it from arbitrary viewpoints or lighting conditions, as we show in Figure 19.

In the fifth column of Figure 18, note that the mouth of the actor is open, resulting in a hole where the teeth should be. Like most methods for facial performance capture, we do not reconstruct the inside of the mouth. Although this effect can be distracting, making the entire face appear different from the reference image, we illustrate that the face reconstruction is still accurate by manually compositing the teeth from the reference frame into the result, as shown in Figure 14. The final result including the teeth now matches the reference frame and is very compelling, indicating an area of future work to simultaneously reconstruct time-varying teeth models with the rest of the face.

We show the versatility of our approach by capturing two other actors, one male and one female. Results are shown in Figure 15 and Figure 16, including the pure geometry result and the high-quality textured rendering. Two extreme zoom renderings are shown in Figure 17(middle and right), using a 10 megapixel texture that we reconstructed from the video images.

[4] www.makehuman.org, www.renderpixie.com

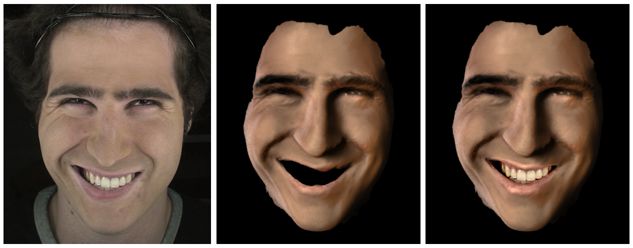

Figure 14: *Adding the teeth creates a compelling result, indicating an area for future research.*

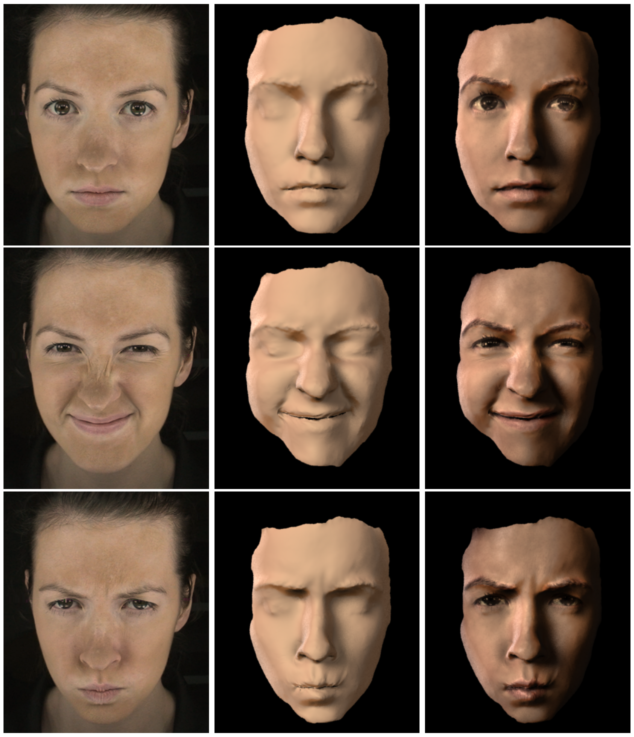

Figure 15: *Capture results for another sequence, including the reference frames (left), pure geometry result (center), and high-quality rendering with texture (right).*

7 Conclusion

In this paper we present a purely passive method for facial performance capture. Unlike previous methods that require markers, face paint, structured light, or expensive hardware, we use only a camera array and uniform illumination. Nonetheless, we are able to reconstruct high resolution, time-varying meshes at 30 frames per second. The absence of markers and structured light allow us to capture detailed, per-frame textures and create high-quality renderings.

One of the keys to our approach is our novel high-resolution acquisition setup. We are able to use natural skin blemishes, hair follicles and pores both for establishing detailed geometric reconstructions, and also for tracking the face over time. Our geometry and texture tracking method is fully automatic, and includes robust temporal drift correction. While the small facial details are visible in the video images, spatial geometry regularization and temporal smoothing prevent us from reconstructing the finest pore-scale geometry. These smoothing steps are required to overcome inaccuracies in the initial reconstructions and in the optical flow vectors.

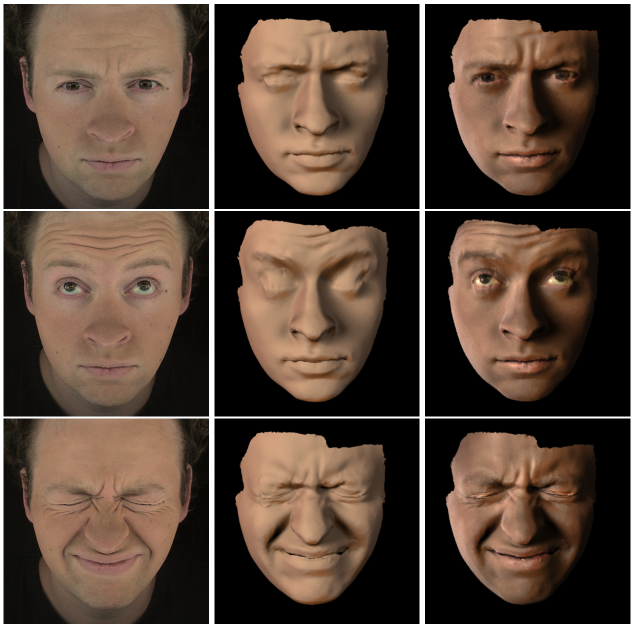

Figure 16: *Capture results for yet another sequence, including the reference frames (left), pure geometry result (center), and high-quality rendering with texture (right).*

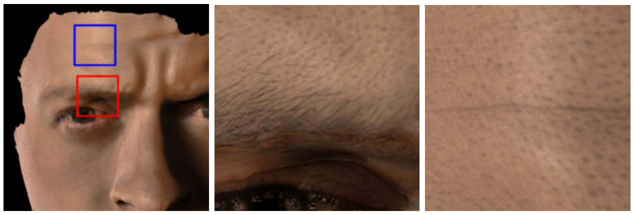

Figure 17: *10 megapixel textures allow extremely close zoom renderings (middle and right).*

However, we do provide much higher resolution than previous passive methods [Li et al. 1993; Essa et al. 1996; DeCarlo and Metaxas 1996; Pighin et al. 1999]. Also note that fine wrinkles and pores can be added to the geometry in a post-process similar to Beeler et al. [2010], and we consider this future work.

Our method is versatile, which we demonstrate by reconstructing three different facial performances given by different actors, and including very different deformations. To our knowledge, this paper presents the first automatic technique to reconstruct high-quality facial performances without the need for markers, face paint, or structured light.

The main limitation of our technique is that very fast motion can lead to incorrect geometry tracking due to motion blur and inaccurate optical flow. Furthermore, since our method processes frames sequentially, an error in tracking could cause an early termination of the face sequence. Additionally, our method is not designed to reconstruct facial hair, and we require manual processing of one frame of the sequence to build the reference mesh.

In the future, we plan to explore methods for automatically completing the face model by capturing the teeth. In addition, correctly capturing the eye geometry, including eyelashes and eyebrows, would produce even more realistic results, particularly for high-quality specular rendering of the eyes.

Acknowledgments

We would like to thank our actors: Ben Cecchetto, Ian South-Dickinson and Jennifer Fernquist. This work was funded in part by the Natural Sciences and Engineering Research Council of Canada and the GRAND Network of Centres of Excellence of Canada.

References

ALEXANDER, O., ROGERS, M., LAMBETH, W., CHIANG, M., AND DEBEVEC, P. 2009. The digital emily project: photoreal facial modeling and animation. In *ACM SIGGRAPH Courses*, 1–15.

BEELER, T., BICKEL, B., SUMNER, R., BEARDSLEY, P., AND GROSS, M. 2010. High-quality single-shot capture of facial geometry. *ACM Trans. Graphics (Proc. SIGGRAPH)*.

BICKEL, B., BOTSCH, M., ANGST, R., MATUSIK, W., OTADUY, M., PFISTER, H., AND GROSS, M. 2007. Multi-scale capture of facial geometry and motion. *ACM Trans. Graphics (Proc. SIGGRAPH)*, 33.

BLANZ, V., BASSO, C., VETTER, T., AND POGGIO, T. 2003. Reanimating faces in images and video. *Computer Graphics Forum (Proc. Eurographics) 22*, 3, 641–650.

BORSHUKOV, G., PIPONI, D., LARSEN, O., LEWIS, J. P., AND TEMPELAAR-LIETZ, C. 2003. Universal capture: image-based facial animation for "the matrix reloaded". In *ACM SIGGRAPH Sketches & Applications*.

BOUGUET, J.-Y. 1999. Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm. Tech. rep., Intel Corporation, Microprocessor Research Labs.

BRADLEY, D., AND HEIDRICH, W. 2010. Binocular camera calibration using rectification error. In *Canadian Conference on Computer and Robot Vision*.

BRADLEY, D., BOUBEKEUR, T., AND HEIDRICH, W. 2008. Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *Proc. CVPR*.

BRADLEY, D., POPA, T., SHEFFER, A., HEIDRICH, W., AND BOUBEKEUR, T. 2008. Markerless garment capture. *ACM Trans. Graphics (Proc. SIGGRAPH) 27*, 3, 99.

BRADLEY, D., ATCHESON, B., IHRKE, I., AND HEIDRICH, W. 2009. Synchronization and rolling shutter compensation for consumer video camera arrays. In *International Workshop on Projector-Camera Systems (PROCAMS 2009)*.

CANNY, J. 1986. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell. 8*, 6, 679–698.

DE AGUIAR, E., STOLL, C., THEOBALT, C., AHMED, N., EIDEL, H.-P. S., AND THRUN, S. 2008. Performance capture from sparse multi-view video. *ACM Trans. Graphics (Proc. SIGGRAPH)*, 98.

DECARLO, D., AND METAXAS, D. 1996. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *CVPR*, 231.

DECARLO, D., AND METAXAS, D. 2000. Optical flow constraints on deformable models with applications to face tracking. *Int. Journal of Computer Vision 38*, 2, 99–127.

ESSA, I., BASU, S., DARRELL, T., AND PENTLAND, A. 1996. Modeling, tracking and interactive animation of faces and heads using input from video. In *Proceedings of Computer Animation*, 68.

FIALA, M., AND SHU, C. 2008. Self-identifying patterns for plane-based camera calibration. *Machine Vision and Applications 19*, 4, 209–216.

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

GUENTER, B., GRIMM, C., WOOD, D., MALVAR, H., AND PIGHIN, F. 1998. Making faces. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 55–66.

KADIR, T., AND BRADY, M. 2001. Saliency, scale and image description. *Int. J. Comput. Vision 45*, 2, 83–105.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21*, 3, 362–371.

LI, H., ROIVAINEN, P., AND FORCHEIMER, R. 1993. 3-d motion estimation in model-based facial image coding. *IEEE Trans. Pattern Anal. Mach. Intell. 15*, 6, 545–555.

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*, 6, 355–372.

MA, W.-C., HAWKINS, T., PEERS, P., CHABERT, C.-F., WEISS, M., AND DEBEVEC, P. 2007. Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. In *Eurographics Symposium on Rendering*, 183–194.

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. Graphics (Proc. SIGGRAPH Asia) 27*, 5, 121.

MOHAMMED, U., PRINCE, S. J. D., AND KAUTZ, J. 2009. Visiolization: generating novel facial images. *ACM Trans. Graph. 28*, 3, 57.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph. 22*, 3, 313–318.

PIGHIN, F. H., SZELISKI, R., AND SALESIN, D. 1999. Resynthesizing facial animation through 3d model-based tracking. In *ICCV*, 143–150.

SHEWCHUK, J. 1996. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, M. C. Lin and D. Manocha, Eds., vol. 1148 of *Lecture Notes in Computer Science*. Springer-Verlag, 203–222.

WANG, Y., HUANG, X., LEE, C.-S., ZHANG, S., LI, Z., SAMARAS, D., METAXAS, D., ELGAMMAL, A., AND HUANG, P. 2004. High resolution acquisition, learning and transfer of dyanmic 3-d facial expressions. In *Computer Graphics Forum*, 677–686.

WILLIAMS, L. 1990. Performance-driven facial animation. *SIGGRAPH Comput. Graph. 24*, 4, 235–242.

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

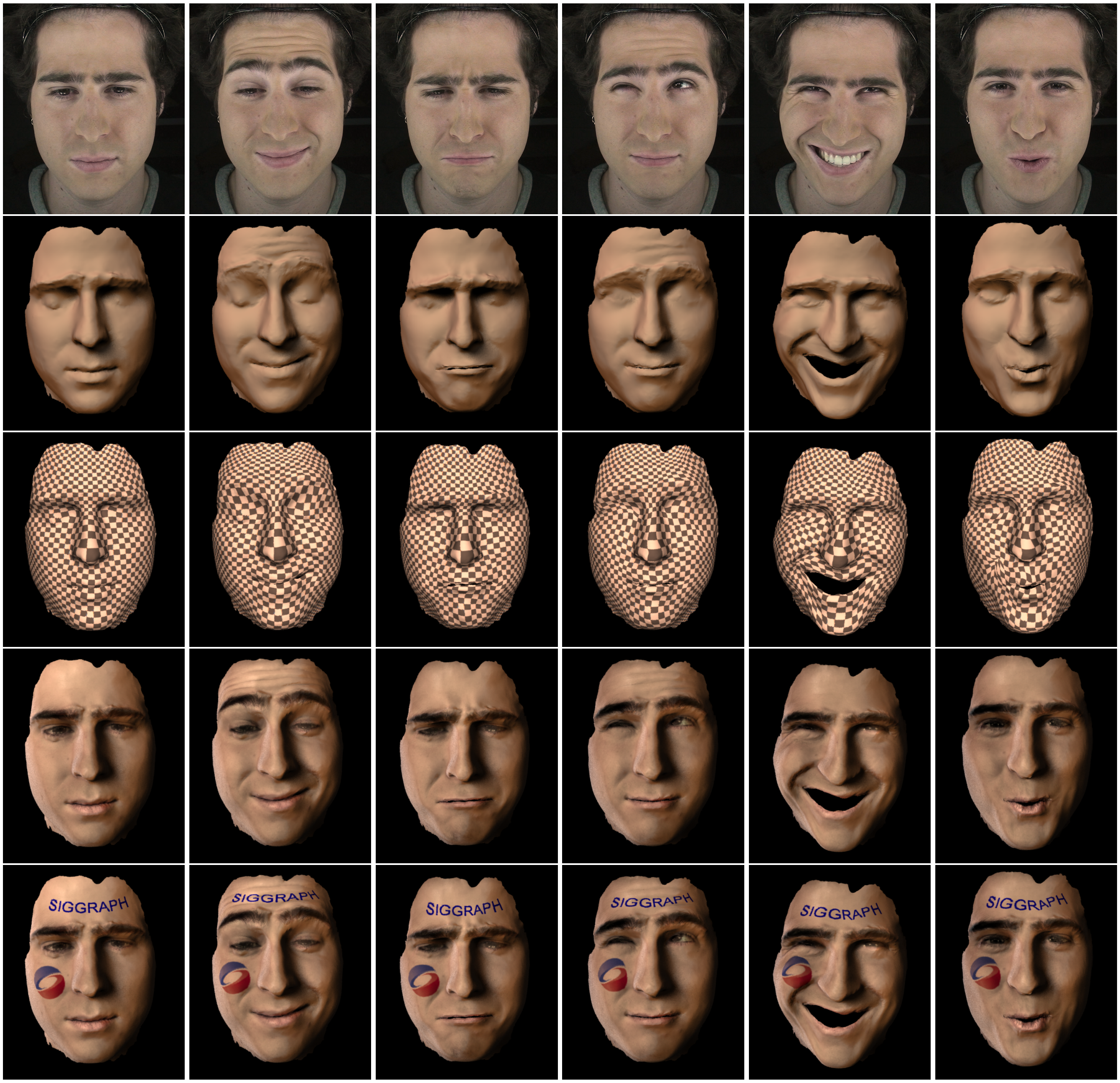

Figure 18: *Capture results for one sequence including the reference footage (top row), pure geometry result (2nd row), skin stretch visualization (3rd row), high-quality rendering with texture (4th row), and virtual makeup (bottom row).*

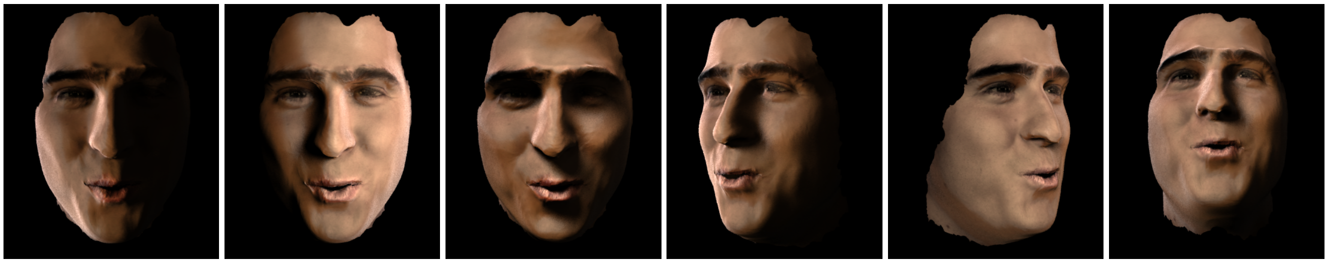

Figure 19: *Realistic renderings under various different illuminations and viewpoints.*