

Marker Optimization for Facial Motion Acquisition and Deformation

Binh H. Le, Mingyang Zhu, and Zhigang Deng, *Senior Member, IEEE*

Abstract—A long-standing problem in marker-based facial motion capture is what are the optimal facial mocap marker layouts. Despite its wide range of potential applications, this problem has not yet been systematically explored to date. This paper describes an approach to compute optimized marker layouts for facial motion acquisition as optimization of characteristic control points from a set of high-resolution, ground-truth facial mesh sequences. Specifically, the thin-shell linear deformation model is imposed onto the example pose reconstruction process via optional hard constraints such as symmetry and multiresolution constraints. Through our experiments and comparisons, we validate the effectiveness, robustness, and accuracy of our approach. Besides guiding minimal yet effective placement of facial mocap markers, we also describe and demonstrate its two selected applications: marker-based facial mesh skinning and multiresolution facial performance capture.

Index Terms—Facial animation, facial deformation, motion capture, marker optimization, thin-shell deformation

1 INTRODUCTION

MOTION capture has been widely used as the standard approach to acquire natural and dynamic motions for computer animation. Among various mocap systems available on the market (e.g., inertial, mechanical, magnetic, and optical) [1], [2], optical motion capture is the most widely used technique due to its flexibility and economy. The optical motion capture techniques can be divided into two categories: *markerless* and *marker based*. The markerless performance capture has many advantages such as make-up free and being able to capture full subject deformation, but it also has many limitations that affect its wide deployment: costly, nonrobust with illumination change, requiring a high data bandwidth, short capture sessions, slow processing, and so on. Although the marker-based technique only captures the motion of limited markers on the subject, it can address all the limitations of the markerless capture. Furthermore, in hybrid mocap tasks (e.g., face + hands, face + full body capture), arguably, a marker-based capture system is still a preferred choice over markerless capture to date. As a result, the marker-based mocap remains one of the most commercially deployed systems worldwide, and it will still be a practical choice in the foreseen future.

Marker placement (i.e., where and how many) is crucial to the usefulness of acquired motion capture data. There are two nontrivial tradeoffs in the design of mocap marker layouts. First, if markers are over-densely placed in a region, besides the unnecessary redundancy in the recorded data, an optical motion capture system may fail to

accurately differentiate individual markers and, thus, impair the data quality. Second, if markers are improperly placed (e.g., too few markers are placed at a region with subtle and dynamic movements), certain movement information may be under-represented in the recorded data.

In full body motion capture, the kinematic structure of the human skeleton can serve as effective guidelines for marker placement. However, such standardized marker layouts for facial motion capture do not exist yet for various reasons. Arguably, the main reasons is that finding optimal marker layouts for facial motion capture is challenging due to the anatomical complexity, large shape variation, and nonrigid deformability of the human face. As a result, researchers and industry practitioners often have to create and use their own empirical (i.e., manually designed) marker setup for facial motion capture. For example, about 50 markers were used for facial capture in the *Avatar* movie. Fig. 1 also illustrates several empirical facial marker layouts including the ones used in the recent work [3], [4] and 68 facial animation parameters (semantically equivalent to facial markers) used in the MPEG-4 facial animation. As shown in this figure, besides the total number of markers, their locations in these empirical layouts vary significantly as well.

Up to date, little, if any, research has been done to quantitatively model and understand the goodness of those numerous empirical facial mocap marker layouts, to the best of our knowledge. In addition, due to the lack of generality and consistency among those empirical facial marker layouts, reusing or combining facial mocap data from different sources (i.e., acquired with different layouts) becomes a nontrivial problem. Indeed, an important yet underexplored problem in marker-based facial motion capture is *what are the optimized (i.e., automatically designed by algorithms) mocap marker layouts for facial motion acquisition and deformation*, although this problem could have a wide range of potential applications. For example, optimized characteristic facial marker layouts will help to increase the

• B.H. Le and Z. Deng are with the Department of Computer Science, University of Houston, 4800 Calhoun Road, Houston, TX 77204-3010. E-mail: {bhle2, zdeng}@cs.uh.edu.

• M. Zhu is with the Nanjing University of Science and Technology, 200 Lingwei St., Nanjing, Jiangsu, China.

Manuscript received 2 Sept. 2012; revised 6 Feb. 2013; accepted 8 May 2013; published online 16 May 2013.

Recommended for acceptance by W. Matusik.

For information on obtaining reprints of this article, please send e-mail to: tcvg@computer.org, and reference IEEECS Log Number TVCG-2012-09-0177. Digital Object Identifier no. 10.1109/TVCG.2013.84.

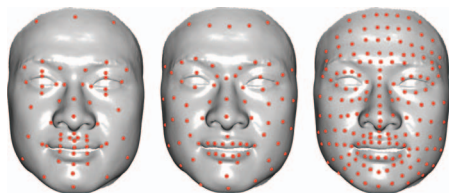


Fig. 1. Several empirical facial mocap marker layouts used in previous work (from left to right: 68 markers in MPEG-4 facial animation standard, 88 markers in [3], 176 markers in [4]). Note that facial markers are illustrated on the same 3D face model by manually transcribing the original marker layouts in those work as accurately as possible.

quality and usefulness of captured facial motion data; thus, many facial animation applications including facial motion acquisition, retargeting, editing, and rigging could potentially benefit from it.

Inspired by the above need and challenge, in this paper, we propose an approach to compute optimized facial marker layouts for facial motion acquisition and deformation. We formulate this problem as optimization of characteristic control points from a set of high-resolution facial pose examples. Specifically, we use pre-acquired high-resolution facial mesh sequences as the ground-truth training examples and minimize their reconstruction errors with respect to a set of characteristic control points. The widely used thin-shell linear deformation model is imposed onto the example pose reconstruction process via optional hard constraints including boundary constraint, symmetry constraint, and multiresolution constraint, to tailor the results toward different applications. Besides validating the effectiveness, robustness, and accuracy of our approach, we also describe its two selected applications: marker-based facial mesh skinning and multiresolution facial performance capture.

2 RELATED WORK

Optical motion capture. As the most widely used motion capture technique, optical motion capture uses photo sensors [5]. The optical motion capture can be further classified to marker-based capture or markerless capture. In a marker-based capture system, a set of cameras are calibrated to track the motion of the markers that are typically either retroreflective (passive markers) or self-illuminated (active markers). At every time frame, the marker positions can be triangulated in the images acquired by more than one camera. Since the set of markers is sparse, finding marker correspondences in different cameras becomes a trivial problem. Due to the advantages in marker segmentation and correspondence finding, the marker-based optical motion capture systems are typically robust. For example, the commercial VICON motion capture system can reconstruct marker positions with a submillimeter accuracy at a high frame rate of up to 240 fps. However, because the number of markers is essentially limited, marker-based capture techniques can only capture surfaces at a low resolution. In the case of facial capture, even when a large number of markers are employed, wrinkles and skin details cannot be accurately captured.

To overcome the resolution limitation, researchers have come up with markerless motion capture techniques

(or called *performance capture*). Unlike the above marker-based motion capture techniques, markerless capture does not require any attachment to the surface of the subject; thus, in theory, continuous surface deformation details can be captured. Typically, a markerless motion capture system needs to deal with the following two problems. 1) The first problem is 3D surface reconstruction, which can be solved by stereo cameras [6], [7], structured light [8], or photometric stereo [9]. Since all of these surface reconstruction techniques rely on the reflectance light from the model, the reconstruction quality is sensitive to the surface material. 2) The second problem is drift correction, which is necessary to maintain the temporal coherence across time frames. This problem is typically solved by video tracking techniques such as optical flow [8], [6], [7].






In addition, due to the large amount of color video data, markerless capture systems typically can only be operated at a low frame rate, for example, 30 [6] and 42 fps [7], compared with 240 fps of marker-based motion capture (e.g., vicon). Because of the above reasons, researchers proposed hybrid solutions to combine the complementary advantages of both marker-based capture and markerless capture to achieve high-fidelity animation [10], [11].

Facial animation. During the past several decades, researchers have conducted extensive research efforts on various aspects of facial animation including face modeling [12], expression retargeting, editing [3], and data-driven facial modeling and animation [13]. In this section, we only briefly review recent efforts in this area. Comprehensively reviewing these efforts is beyond the scope of this paper; interested readers are referred to the recent survey [14].

Researchers developed a variety of techniques to automatically or semiautomatically transfer facial expressions from one face model to another [15], [16], [17], [18], [19] or to blendshape faces [20], [21]. Also, various interfaces and approaches have been developed to efficiently edit existing facial animation sequences by learning statistical models from acquired face data sets [22], [23], [24], [4]. With the increase of acquired high-fidelity facial motion data sets, many data-driven facial animation approaches have been developed for blendshape face rigging [25], [26] or to generate realistic speech animations for novel typed or spoken input, based on pre-collected facial motion data sets [27], [28], [29], [30], [31].

Mesh compression. The problem formulation in this work shares certain similarities with existing mesh compression techniques [32], [33]. Both of the mesh compression methods [32], [33] compress mesh geometry by extracting a set of control points from the mesh vertices and then using the positions of the control points to resolve the positions of all mesh vertices in a least squares sense. While the least-squares meshes technique [32] finds the optimal control points from a single static mesh, the motion-sensitive anchor identification of least-square meshes [33] can utilize a mesh sequence to detect the optimal control points. In this work, the role of control points is quite similar to that of the markers because both of them can be used to encode the 3D mesh geometry. However, there is a major difference: the markers are constrained to be on the facial mesh surface, while the control points are not necessary to be on the

TABLE 1
Statistics Information of the Five Used High-Resolution
3D Face Mesh Sequence Data Sets

					
Name	Model #1	Model #2	Model #3	Model #4	Model #5
Vertex # (Org)	23725	513117	532060	314702	474445
Triangle # (Org)	46853	1023594	1061381	627926	946683
Vertex # (Sim)	-	25342	25340	25301	25274
Triangle # (Sim)	-	49999	49999	50000	49999
Frame #	384	361	77	226	321

“Org” denotes the original models, and “Sim” denotes the simplified models.

surface. For this reason, the set of control points cannot always accurately recover facial animations, as demonstrated in Section 6.3.

3 FACIAL MOTION DATA SETS

In this work, we use five high-resolution 3D face mesh sequences acquired by the cutting-edge performance capture systems [8], [6]. For the sake of convenience, a unique name (i.e., “model #”) is assigned to each data set in this writing (refer to Table 1). Among these data sets, model #1 was acquired by Zhang et al. [8], and models #2 to #5 were acquired by Bradley et al. [6]. The face meshes in every sequence share the same topology, and the five mesh sequences are regarded as the ground-truth facial deformation data. We also denote the combination of all the five data sets as #All. Note that since the original meshes of models #2 to #5 are overdense, compared with model #1, we reduced the number of triangles in each model to 50,000 using the quadric error metric based simplification algorithm [34], and we only simplified the first frame and then calculated other frames to ensure topology consistency across frames. This decimation introduced an error of 0.3 mm on average. Since this amount of error cannot be captured by linear deformation models, this decimation would not affect final results.

4 METHODOLOGY

4.1 Assumption and Problem Analysis

Since the ultimate goal of motion capture is to record the performance information of a subject as much as possible, we can use this criterion to quantify the goodness of a facial marker layout. Fortunately, recent facial performance capture advances allow us to faithfully record minute facial movements. The output of these facial performance capture systems is high-resolution 3D facial mesh sequences with strict spatial-temporal coherence (i.e., vertex-to-vertex correspondence across frames).

Although marker-based facial mocap can only accurately record marker displacements, facial marker displacements and marker-driven linear deformation models have demonstrated their competence to model large-scale facial motion [3], [11]. Therefore, we can identify the optimized facial marker layouts by minimizing the deformation error if the marker-driven linear facial deformation model is used to

reconstruct the ground-truth facial motions, acquired by the recent cutting-edge facial performance capture systems [8], [6], [7].

Constraints. To constrain the marker-driven linear deformation model in our particular problem, we also need to introduce three optional constraints, namely, *symmetry constraint*, *multiresolution constraint*, and *boundary constraint*. Through flexible combinations of the three constraints, our approach can optimize and tailor the resultant facial marker layouts toward different applications.

Definitions of the three optional constraints are described below. 1) *Symmetry constraint.* The shapes of human faces are approximately left-right symmetric in general; thus, the optimized marker layouts are expected to be symmetric. 2) *Multiresolution constraint.* Depending on different applications, we may want to optimize layouts with different numbers of markers. In particular, the optimized layout with a lower resolution is a subset of the optimized layout with a higher resolution (i.e., the expected number of markers is larger). This multiresolution constraint would enable users to incrementally add markers to the face in marker-based facial mocap applications, while ensuring the backward compatibility of the previously acquired motion data. 3) *Boundary constraint.* Facial movements typically only happen in the front and certain side regions. Thus, there exists a boundary on the face to separate the deformation area and the nondeformation area. To avoid the optimized markers being distributed on the unnecessary nondeformation area of the face, we need to put a number of “virtual fixed markers” on the boundary, called the boundary constraint in our approach.

In follow-up Sections 4.2, 4.3, 4.4, and 4.5, we will first formulate and solve the optimized marker layout problem without considering the above three constraints. Then, we will describe how to incorporate the constraints to our approach in Section 4.6. Finally, in Section 4.7, we will provide a solution for combining the input data from multiple subjects.

4.2 Problem Formulation

Let F be the number of frames in a facial mesh animation sequence (e.g., one of the five ground-truth facial mesh data sets described in Section 3), and N be the number of vertices in each mesh. Assuming $p_i \in \mathbb{R}^3$ is the 3D coordinate of vertex i in the rest pose, and $v_i^{(t)} \in \mathbb{R}^3$ is the coordinate of the i th vertex at the t th frame. The facial mesh sequence can be represented as a displacement matrix, $X \in \mathbb{R}^{3F \times N}$, where three consecutive rows represent the displacements of the N vertices in one time frame and each column represents the displacements of one vertex over F time frames. The matrix X has the following form:

$$X = \begin{bmatrix} v_1^{(1)} - p_1 & v_2^{(1)} - p_2 & \cdots & v_N^{(1)} - p_N \\ v_1^{(2)} - p_1 & v_2^{(2)} - p_2 & \cdots & v_N^{(2)} - p_N \\ \vdots & \vdots & \ddots & \vdots \\ v_1^{(F)} - p_1 & v_2^{(F)} - p_2 & \cdots & v_N^{(F)} - p_N \end{bmatrix}.$$

Let M be the number of markers in the optimized marker layout that we aim to find. Since the markers can only be located at the vertices of the facial mesh; thus, the

M -markers layout can be represented as a set of M indices $\mathcal{H} = \{h_1, h_2, \dots, h_M\}$, where $\mathcal{H} \subset \{1, 2, \dots, N\}$. Suppose this marker layout is put on to the input facial mesh, the displacements of the markers over all the frames in X can be represented as a matrix $H \in \mathbb{R}^{3F \times M}$, which is similar to the form of X . In the H matrix, each column represents the displacements of one marker over F time frames. At any time frame, the displacement of marker j equals to that of vertex h_j , described as follows:

$$H_{i,j} = X_{i,h_j}, \forall i, j. \quad (1)$$

Now assuming the marker displacement matrix H is available, we can reconstruct the displacements of the facial mesh animation sequence X . Let $X' \in \mathbb{R}^{3F \times N}$ be the reconstructed displacement matrix. If we employ a linear deformation method for reconstruction, the displacement matrix can be described as

$$X' = HW^T, \quad (2)$$

where $W \in \mathbb{R}^{N \times M}$ is the weight matrix, and $W_{i,j}$ denotes the weight (or the influence) of marker j to vertex i . The weight matrix W is calculated from the marker layout \mathcal{H} by employing a linear deformation algorithm. In this work, we choose the *thin-shell deformation model* [10], [3], [35] to demonstrate our facial marker layout optimization framework.

It is noteworthy that the thin-shell deformation model can be conceptually regarded as a special case of linear blend skinning (LBS), where the bone rotations are not considered, and the skinning weights are calculated by minimizing the surface energy. In general, other LBS models without bone rotations and a known skinning weights calculation method, for example, radial basis functions, can also be used within our framework. This class of LBS can model facial deformation quite well in the case that the global head rotation is removed. In this work, we do not consider the blendshape model, because this will require more complicated analysis on the selection of blendshape bases.

4.2.1 Thin-Shell Deformation Model

In this work, we use the thin-shell deformation model [10] to deform the rest facial pose via marker displacements. More details of this model can be found in [10], [35]. For the sake of clarity and completeness, we briefly describe it in this section.

Basically, in this model, the elastic energy of a surface is minimized while a subset of the surface is constrained. Here, the surface of the deformable object is limited to a two-manifold surface, or the thin shell. The energy of the surface consisting of two terms, namely, the *stretching* energy and the *bending* energy, is used to quantify the surface deformation from the rest pose. Minimizing this surface energy yields the following Euler-Lagrange PDE:

$$-k_s \Delta d + k_b \Delta^2 d = 0, \quad (3)$$

where d denotes the displacement on the surface from the rest pose, Δ is the Laplace-Beltrami operator, and k_s and k_b are the stretching coefficient and bending coefficient, respectively. If a subset of the surface is constrained (i.e., its

displacement is given), the whole surface displacement in (3) can be solved.

In the case that the surface of the deformable object is represented as a triangle mesh, the above Euler-Lagrange PDE (3) can be discretized by the cotangent weights [36], which yields the following linear system of equations:

$$(-k_s \mathcal{L} + k_b \mathcal{L}^2)d = 0, \quad (4)$$

where d , k_s , and k_b have the same meanings as above, and \mathcal{L} is the Laplacian of the mesh. \mathcal{L} can be calculated via a normalization weight matrix \mathcal{M} and an edge weight matrix \mathcal{L}_s (refer to (5)), as described in the work of [35]. Note that in (5), \mathcal{M} is a diagonal matrix and \mathcal{L}_s is a symmetric, seminegative definite matrix:

$$\mathcal{L} = \mathcal{M}^{-1} \mathcal{L}_s. \quad (5)$$

We can premultiply (4) by \mathcal{M} , which yields the following symmetric system:

$$\underbrace{(-k_s \mathcal{L}_s + k_b \mathcal{L}_s \mathcal{M}^{-1} \mathcal{L}_s)}_A d = 0. \quad (6)$$

Then, the constraints on vertices are imposed as hard constraints by moving each column corresponding to a constrained vertex to the right-hand side and removing the corresponding row from the linear system [35]. Since this solution changes the structure of the A matrix, it is only suitable if the constrained vertices are kept unchanged. Unfortunately, our formulated optimization problem requires probing different marker layouts (that is, different constrained vertices), which makes this solution inefficient. By contrast, we propose a modification in the A matrix: instead of removing rows and columns, only the values of those rows and columns are changed.

Recalling our problem, in the set of N mesh vertices, M markers $\mathcal{H} = \{h_1, h_2, \dots, h_M\}$ play the role as the constrained vertices, where h_j is the index of the vertex at which the j th marker is located. Constraining (6) to the marker set \mathcal{H} yields the following system:

$$\hat{A}d = \hat{B}d^{\mathcal{H}}, \quad (7)$$

where: d_i is the displacement of vertex i ,

$d^{\mathcal{H}}_j$ is the displacement of marker j ,

$$\hat{A}_{i,j} = \begin{cases} 1 & \text{if } i = j \text{ and } i, j \in \mathcal{H}, \\ 0 & \text{if } i \neq j \text{ and } i \in \mathcal{H} \text{ or } j \in \mathcal{H}, \\ A_{i,j} & \text{otherwise,} \end{cases} \quad (8)$$

$$\hat{B}_{i,j} = \begin{cases} 1 & \text{if } i = h_j, \\ 0 & \text{if } i \in \mathcal{H} \text{ and } i \neq h_j, \\ -A_{i,h_j} & \text{otherwise.} \end{cases} \quad (9)$$

This linear system can be achieved from a similar linear system in [35] by adding M equations of the form $d_{(h_j)} = (d^{\mathcal{H}})_j$. Thus, the deformation results solved from the two systems are identical. Let

$$W = \hat{A}^{-1} \hat{B}. \quad (10)$$

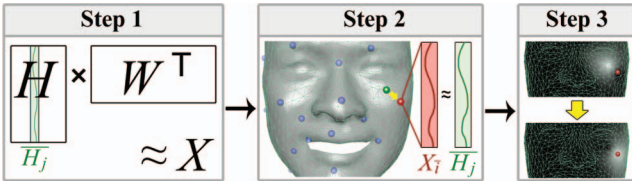


Fig. 2. Schematic illustration of our marker refinement process. Step 1 finds the expected trajectory of the marker without requiring it to follow the displacements of any mesh vertices. Step 2 relocates the marker to the vertex, which has the best-matched displacement with the expected marker trajectory in step 1. Step 3 updates the weight matrix W corresponding to the new layout.

Then, the solution of the above linear system (7) is $d = \hat{A}^{-1} \hat{B} d^H = W d^H$. Note that here the matrix W is the same weight matrix in (2).

Putting everything altogether, we can find the optimized marker layout by minimizing the error of the reconstructed displacements in (2), subject to the constraints in (4) and (10). In particular, we formulate our facial marker layout optimization problem as the following constrained least squares:

$$\min_{\mathcal{H}=\{h_1, \dots, h_M\}} E = \min_{\mathcal{H}} \|X - HW^T\|_2^2, \quad (11a)$$

$$\text{s.t.}: H_{i,j} = X_{i,h_j}, \forall i, j, \quad (11b)$$

$$W = \hat{A}^{-1} \hat{B}, \hat{A} \text{ and } \hat{B} \text{ calculated by (8) and (9)}. \quad (11c)$$

4.3 Algorithm Overview

Our core idea of optimizing the above objective function, (11), is based on the block coordinate descent algorithm [37]. Basically, it first divides the set of variables into several blocks. Then, in an iterative manner, the objective function is alternatively optimized with respect to one block while other blocks are kept unchanged. In our algorithm, we consider each marker as a block.

Within this framework, our problem then becomes how to refine the position of the j th marker in a given layout to get a smaller reconstruction error. We do this in an expectation-correction fashion. At the expectation step, we first relax the constraint, (11b), to find the expected trajectory of the marker j , without requiring it to follow displacements of any mesh vertices. Then, at the correction step, we find a vertex \bar{i} which has the best-matched displacement with the expected marker trajectory to correct the constraint (11b). Finally, we relocate the marker j to the location of vertex \bar{i} and update the weight matrix corresponding to the new layout.

The overview of our optimization algorithm is described in *Algorithm 1*. The algorithm starts by initializing a randomly generated marker layout (line 1) and solves the weight matrix corresponding to this layout (line 2). The weight matrix W is solved through the thin-shell deformation model (detailed in Section 4.5). Then, in each iteration of the main loop, we refine the markers one by one.

Algorithm 1. Facial Marker Layout Optimization.

Input: X, A, M

Output: $\mathcal{H} = \{h_1, h_2, \dots, h_M\}$ s.t. (11)

1: Initialize a marker set \mathcal{H}

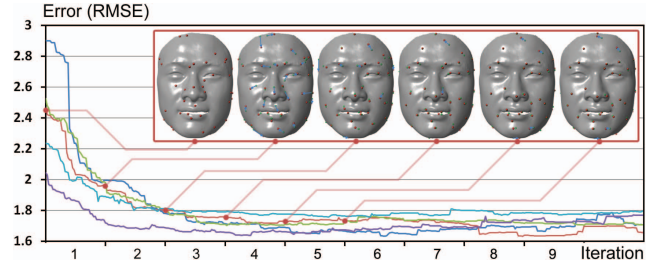


Fig. 3. RMSE error plotting of our algorithm running on model #1 five times with different initial layouts when the number of markers is set to 40. Although our algorithm cannot guarantee a strict error descent, the RMSE error is generally decreased quickly in our experiments such as 3-5 iterations. The marker layouts in five iterations of one trial (red) are showed at the upper right corner. The red markers illustrate the layout after the current iteration and the green markers illustrate the layout after the previous iteration.

- 2: Solve the weight matrix W according to \mathcal{H}
- 3: **repeat**
- 4: **for** each marker $j \in \{1, 2, \dots, M\}$ **do**
- 5: **Step 1:** Calculate the expected trajectory \bar{H}_j
- 6: **Step 2:** $h_j \leftarrow \bar{i} = \arg \min_i \|X_i - \bar{H}_j\|_2$
- 7: **Step 3:** Update the weight matrix W for h_j change
- 8: **end for**
- 9: **until** Convergence OR a maximum of iterations are reached
- 10: **return** \mathcal{H}

For each marker j , our marker refinement process includes the following three steps (line 5 to line 7), as illustrated in Fig. 2. 1) We first calculate its expected trajectory \bar{H}_j to reduce the reconstruction error E (line 5). \bar{H}_j is calculated by solving a linear least squares problem, as detailed in Section 4.4. 2) We update the layout by assigning marker j to the vertex \bar{i} with the trajectory closest to the expected trajectory \bar{H}_j (line 6). Here, we simply use the euclidean distance to measure the closeness of two trajectories. The index of the vertex with the closest trajectory is determined as $\bar{i} = \arg \min_i \|X_i - \bar{H}_j\|_2$. If another marker has already been located at vertex \bar{i} , we put marker j at a random vertex $i \notin \mathcal{H}$. 3) After the marker j is relocated, we update the weight matrix W according to the change of h_j (line 7). Since there is only one change of h_j in the marker set, we can improve the performance by reusing the old layout to update W . This weight matrix updating is detailed in Section 4.5.

Our marker refinement process relies on the calculation of the expected marker trajectory, but it cannot guarantee a strict descent of the reconstruction error E . Thus, we cannot use the value of E to check the convergence of the algorithm. Instead, we stop the algorithm if there is no change on h_j for all $j = 1, \dots, M$ after certain iterations. Although the convergence of our algorithm is difficult to be mathematically proved, the reconstruction error E is generally decreased in our experiments. Fig. 3 shows the change of the root mean square error (RMSE), $\sqrt{E/(N \cdot F)}$, during the optimization process. Specifically, we ran our algorithm on the model #1 five times with different initial layouts when the number of markers was set to 40. The error curves illustrate the RMSE change at every marker refinement step (note that each iteration includes 40 steps of

marker refinement in this experiment). From this figure, we can also observe that our algorithm typically converges quickly, such as within three to five iterations.

4.4 Calculating the Expected Marker Trajectory

Suppose $M - 1$ markers in the layout (i.e., only excluding the j th marker) are unchanged, we want to calculate the expected trajectory \overline{H}_j for the j th marker so that the reconstruction error is decreased. Let A_j denotes the j th column of matrix A . We calculate \overline{H}_j by minimizing the objective function E in (11) with respect to H_j while relaxing the constraint $H_j = X_{h_j}$:

$$\overline{H}_j = \arg \min_{H_j} E = \operatorname{argmin}_{H_j} \|X - HW^T\|_2^2.$$

We can expand the objective function as follows:

$$\begin{aligned} E &= \|X - HW^T\|_2^2 \\ &= \left\| X - \sum_{k=1, k \neq j}^M H_k(W_k)^T - H_j(W_j)^T \right\|_2^2. \end{aligned}$$

Let $R = X - \sum_{k=1, k \neq j}^M H_k(W_k)^T$, the above equation yields

$$E = \|R - H_j(W_j)^T\|_2^2.$$

Since E is a quadratic function with respect to H_j , we can minimize E by solving the equation of the gradient vector:

$$\nabla E = 0 \Leftrightarrow (W_j)^T W_j H_j - (W_j)^T R = 0.$$

By solving the above equation, we can have the expected trajectory as follows:

$$\overline{H}_j = \frac{(W_j)^T R}{(W_j)^T W_j}. \quad (12)$$

4.5 Updating Weight Matrix

We can directly calculate the weight matrix W in (10) by solving the following linear system:

$$\hat{A}W = \hat{B}. \quad (13)$$

Here, \hat{A} and \hat{B} can be calculated from the matrix A and the current marker layout \mathcal{H} by (8) and (9). Since the matrix A is sparse, symmetric, and positive definite, \hat{A} also holds the same properties. Thus, the linear system, (13), can be directly solved by the Cholesky Decomposition [35]. This direct solution involves two steps: the first step is to perform Cholesky decomposition on matrix \hat{A} , and the second step is to solve multiple right-hand side systems, where each right-hand side corresponds to a column of \hat{B} . In the above Algorithm 1, we apply this direct solution to the initialization step (line 2).

At the weight matrix updating step (line 7 in Algorithm 1), since only one marker is changed, some calculations can be saved. Suppose before the layout updating step (line 6 in Algorithm 1), the j th marker is located at vertex h_j , and after this update the new location of the j th marker is vertex h'_j , we need to modify the matrix \hat{A} and \hat{B} to fulfill (8) and (9). The modifications of the matrix \hat{B} involve changing rows h_j and h'_j , which can be done straightforwardly. For the matrix \hat{A} , its

TABLE 2

Performance Comparison between the Direct Solver by Cholesky Decomposition (Abbreviated as *Origin*) and Our Modification on the Decomposition Matrix (Abbreviated as *Ours*)

Thread #	Averaged Running Time (milli-seconds)					
	4		8		16	
Method	Origin	Ours	Origin	Ours	Origin	Ours
$M = 40$	794	303	657	154	582	89
$M = 60$	952	449	724	230	634	121
$M = 80$	1146	611	815	313	667	154

The used test data set is the model #1 ($N = 23,725$ vertices). M denotes the number of markers. The running time is reported as the average value of 10 runs on the 2.4-GHz 16-core Xeon processor.

modifications involve changing both rows and columns h_j and h'_j , followed by the Cholesky decomposition of the new matrix. The Cholesky decomposition of a sparse matrix can be effectively updated (e.g., modifying a column/row) with the time proportional to the number of changed nonzero elements [38]. In our implementation, we only perform the numerical modification while the symbolic decomposition is kept unchanged to avoid unnecessary overhead.

After the modifications of \hat{B} and the Cholesky decomposition of \hat{A} , we do the second step of solving multiple right-hand side systems in parallel. Although the multiple right-hand solver can be performed individually in multiple threads, the Cholesky decomposition on the sparse matrix cannot be performed in the same way [38]. Thus, employing our proposed modifications on the decomposition matrix will significantly improve the bottleneck performance. As shown in Table 2, our modifications run several times faster than the direct solver, and the performance of our approach scales well when the number of processing units is increased. In our implementation, we used the CHOLMOD open source library for sparse Cholesky factorization and update [38].

4.6 Handling Constraints

Symmetry constraint. We impose the symmetry constraint onto the marker layout by pairing the markers in \mathcal{H} , for example, always keeping marker $2j - 1$ and $2j$ as a symmetric pair for $j = 1, \dots, M/2$, assuming M is even. Then, we only need to do the following modifications in the optimization Algorithm 1 to impose the symmetry constraint through marker pairing. 1) The first modification is to mirror the input data to ensure its symmetry, because the original captured subject's facial motions may not be perfectly symmetric. We perform the facial mesh mirroring by manually specifying a mirror plane \mathcal{P} in the rest pose and then adding a symmetric displacement copy for each example pose. Let i_p be the mesh vertex index in the rest pose, which is the closest to the \mathcal{P} -mirrored location of the i th vertex. For each row t (representing one mesh frame in the sequence) in the displacement matrix X , we add a new row t' , where $X_{t',i} = S(X_{t,i}, \mathcal{P}), \forall i = 1, \dots, N$. Here, $S(\cdot, \mathcal{P})$ denotes the mirror transformation through the plane \mathcal{P} . Since the resolution of the facial mesh is high, we can safely assume the mirrored data can approximate symmetric actions. 2) The second modification is to only update markers with odd indices in the layout (i.e., markers with indices in the form of $2j - 1$). For instance, if after a marker refinement step, the new location of marker $2j - 1$ is $h_{2j-1} = i$, then we

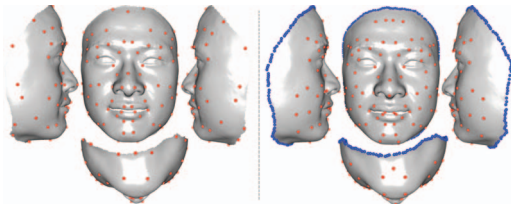


Fig. 4. A boundary constraint comparison (80 markers, model #1): without the boundary constraint (left) and with the boundary constraint (right). The boundary vertices are blue colored.

accordingly set the new location of its paired marker $h_{2j} = i_p$. Also, at the initialization step in Algorithm 1, we will ensure the symmetry of the initial layout.

Boundary and multiresolution constraints. We impose both the boundary and the multiresolution constraints onto our algorithm by introducing a set of predefined markers, \mathcal{S} . The predefined set of markers is kept unaltered during the optimization process. In the Algorithm 1, we can keep \mathcal{S} by simply not updating those markers. Then, the boundary constraint can be imposed by adding vertices on the face boundary to \mathcal{S} . To improve the performance, we can uniformly sample the boundary vertices instead of adding all of them to \mathcal{S} . The multiresolution constraint can be imposed by adding markers in the lower resolution layout to \mathcal{S} when the higher resolution marker layout is being solved by Algorithm 1.

4.7 Combining Input from Multiple Subjects

We can reduce the data dependence of our method by combining input from multiple captured sessions and from multiple subjects. While using the data sets from multiple captured sessions can be handled easily with proper setup, using data sets captured from multiple subjects requires accurate correspondences between different data sets. We employ the nonrigid facial registration framework proposed by Mao et al. [39] to build these dense correspondences. Let D be the number of data sets, we first register the rest pose of the data sets #2 ... #D to the data set #1, the registration maps each vertex in the data sets #2 ... #D to one vertex in the data set #1. With this mapping, we update the marker layout (Step 2 in Algorithm 1) with respect to the vertex positions on the data set #1 while the expected trajectories (Step 1) and the weight matrices (Step 3) are calculated per subject.

5 RESULTS

We conducted various experiments to test our approach on the five facial mesh sequence data sets (described in Section 3). The key parameters of the thin-shell deformation model [3] were empirically chosen as follows: $k_S = 1$ and $k_B = 10$. The marker layouts are generated by running our algorithm 10 times with different initializations and taking the best result. In each trial, the maximum number of iterations is set to be 10. The RMSE error is used as our quantitative metric in our experiments. Demo video (mpeg4 video) can also be accessed at the following link: <http://www.youtube.com/watch?v=7Gfo2teAlqc>.

Fig. 4 shows a comparison with/without the boundary constraint. Fig. 5 shows comparison results with/without the

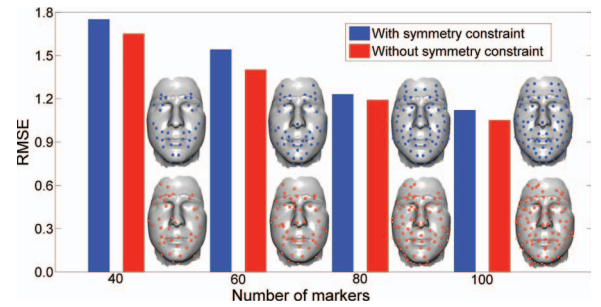


Fig. 5. Result comparison of the model #2 with/without imposing the symmetry constraint to our approach. The resultant optimized marker layouts are also visualized.

symmetry constraint. As shown in Fig. 5, besides the RMSE error, the optimized layouts with the symmetry constraint are sufficiently close to those without the symmetry constraint, with respect to the markers' locations. We use the boundary constraint and the symmetry constraint in our remaining experiments unless explicitly mentioned.

Fig. 6 shows the visualized vertex deformation errors driven by different numbers of optimized markers. We calculate the error of each vertex as the euclidean distance between its ground-truth position (available in the original data sets) and its deformed position.

Fig. 7 shows comparison results with/without the multiresolution constraint. When the resolution is increased, additional new (blue) markers are added. Meanwhile, the RMSE error is decreased accordingly. Also, compared with the cases without the multiresolution constraint, we can observe that the multiresolution results have slightly larger RMSE errors, because the multiresolution layouts are optimized with additional constraints, that is, the

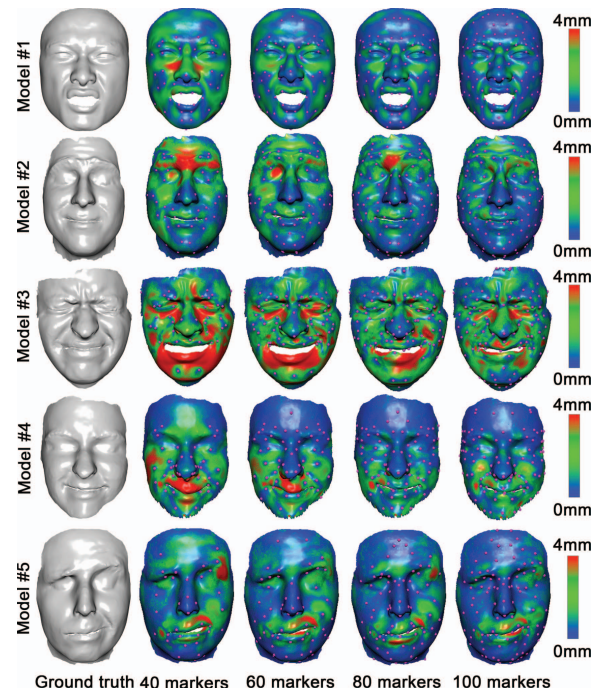


Fig. 6. Visualized vertex deformation errors of all the five face data sets when the number of optimized markers is increased (from 40, 60, 80, to 100). The used error unit is mm.

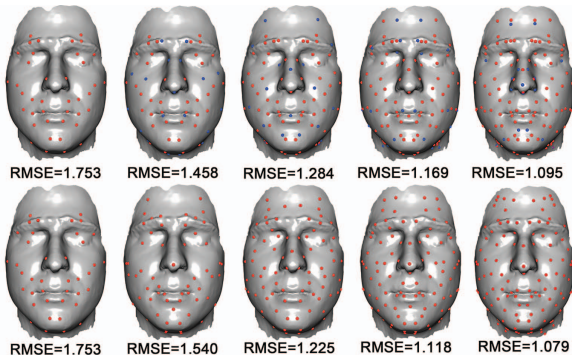


Fig. 7. Optimized marker layout comparison: with the multiresolution constraint (top row) and without the multiresolution constraint (bottom row) on model #2. The red markers are inherited from the left-neighbor lower resolution layout and the blue ones are the new added markers.

markers (red) inherited from the left-neighbor lower resolution layout.

Fig. 8 shows the marker layouts obtained by combining all the five data sets (#All). These layouts could be used as the practical guidelines for the placement of facial markers.

We implemented our algorithm on an off-the-shelf computer, where the GPU is only used to perform large matrix operations and RMSE calculations. Table 3 shows the obtained computational time when our algorithm is applied to the five data sets. Note that, essentially, our approach is a discrete, multivariable optimization algorithm, where each marker is a variable. Therefore, in theory it is difficult to guarantee our approach can reach the global optimum. To the end, running more iterations in our algorithm is necessary to optimize the layout when the number of markers is increased.

6 VALIDATIONS

6.1 Validation via Simulation Data

We validated our marker optimization Algorithm 1 via simulation. To generate the simulation data, we first randomly distribute M markers on the rest pose of the model #1 and also apply uniform random displacements of the M markers in the range of $[-10, 10]$ mm]. Based on the rest pose and the generated marker displacements, we use the thin-shell deformation model to generate the displacements of all the vertices. At the validation step, we use our Algorithm 1 to inversely solve the optimized marker layout based on the simulation data (as the input). Ideally, if the exact marker layout can be found by our algorithm, the RMSE error metric (11) will be zero. On the other hand, since our algorithm cannot always guarantee its convergence at

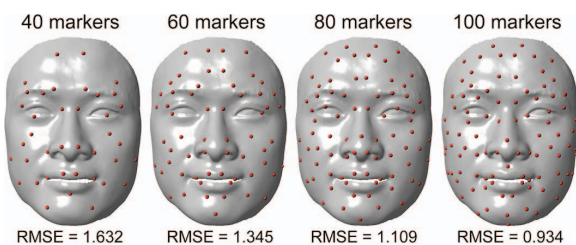


Fig. 8. Some facial marker layouts obtained by our approach through combining all the five face data sets.

TABLE 3

The Obtained Computational Time, Measured in Minutes, of Our Approach when It Was Applied to the Six Face Data Sets

Mkr #	Model #1	Model #2	Model #3	Model #4	Model #5	All Models
40	73.8	68.3	34.4	51.9	59.8	285.7
60	104.5	92.1	52.3	75.4	89.2	408.7
80	145.2	137.4	77.1	101.6	132.7	582.5
100	164.3	161.6	97.1	129.4	150.5	702.3

The hardware configuration of the used computer is: Intel Xeon 2.4-GHz 16-core CPUs, and a NVIDIA Tesla C1060 240-core GPU.

the global optimum in theory; thus, we only expect the RMSE error is close to zero, and the marker layout solved by our algorithm is sufficiently close to the one originally used for generating the simulation data.

Fig. 9 shows the simulation-based validation results with three different numbers of markers ($M = 40, 60,$ and 80). In this experiment, we ran our algorithm 20 times, each time with a maximum of 10 iterations, and the best results are illustrated. From this figure, we can see the RMSE errors are measurably small (0.167 to 0.228), and the marker layouts solved by our optimization algorithm are very similar to the original ones. The fewer the markers, the more accurate the simulation result is (i.e., a smaller RMSE error and a more similar layout). The main reason is if the number of markers is increased, then the optimization problem will be more complex (i.e., more variables). Therefore, the chance of finding the global optimum (i.e., the original layout) by our algorithm will be smaller, and the RMSE error will be larger.

6.2 Cross-Model Validations

We also performed cross-model validation experiments in a leave-one-out fashion. Basically, in each experiment, we use four data sets (out of the total five available data sets, refer to Section 3) to find the optimized marker layout; this layout is tested on the remaining data set. The dense correspondences between different data sets are calculated by using the work of [39], as described in Section 4.7. The deformed RMSE errors with different test models and different numbers of markers are presented in Fig. 10. The results demonstrate that the marker layouts optimized by our algorithm can be effectively used to guide the manual marker placement on another subject in marker-based facial mocap applications.

6.3 Comparisons with Baseline Approaches

We also compared our approach with the baseline approaches that includes least-squares meshes approach [32]

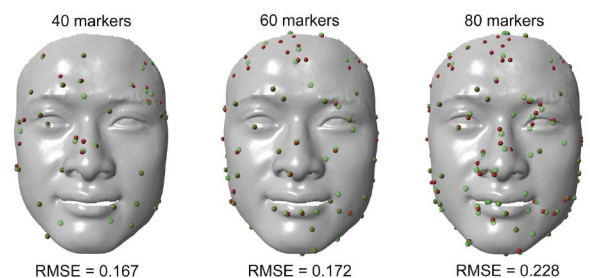


Fig. 9. The simulation-based validation results on model #1. Red dots illustrate the original randomly distributed M markers, and green dots illustrate the markers solved by our optimization algorithm based on the simulation data.

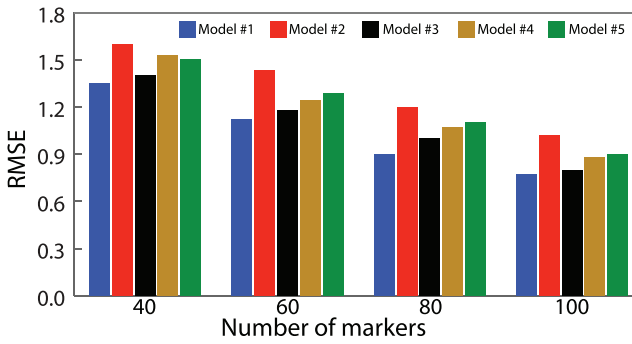


Fig. 10. Leave-one-out cross-model validation results of applying the optimized marker layouts from four subjects to the remaining subject’s ground-truth facial mesh sequence data set.

(abbreviated as *LS-meshes* in this writing), motion-sensitive anchor identification of least-square meshes approach [33] (abbreviated as *MSLS-meshes* in this writing), and three selected empirical facial marker layouts used in the MPEG-4 facial animation and the work of [3], [4] (refer to Fig. 1). The LS-meshes approach is chosen since it can also be used to extract a user-specified number of characteristic control points from a single mesh. Specifically, in the comparison, the LS meshes approach is applied to the first frame of the mesh sequence and its ω parameter is set to 20. The MSLS-meshes approach uses LS-meshes approach to reconstruct the mesh and calculate the anchor points based on clustered teleconnection analysis on the motion sequence. In our experiment, the cluster number is set to 6, λ and μ are set to 0.1 and 0.5, respectively, as reported in the original paper [33]. For each mesh data set, the same boundary constraint is used for all the methods for the sake of a fair comparison. It is noteworthy that different from the LS-meshes approach that only utilizes the first frame, the MSLS-meshes approach [33] utilizes all the facial mesh frames in the data set.

Fig. 11 shows the comparison of the vertex deformation errors among our approach, the LS-meshes approach and

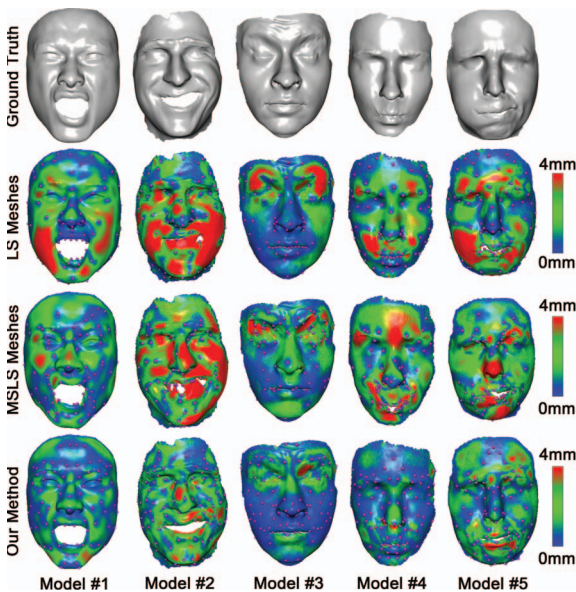


Fig. 11. Comparisons among our approach, the LS-meshes approach [32] and the MSLS-meshes approach [33] (80 optimized markers on all the data sets). The vertex deformation errors (unit: mm) are color coded.

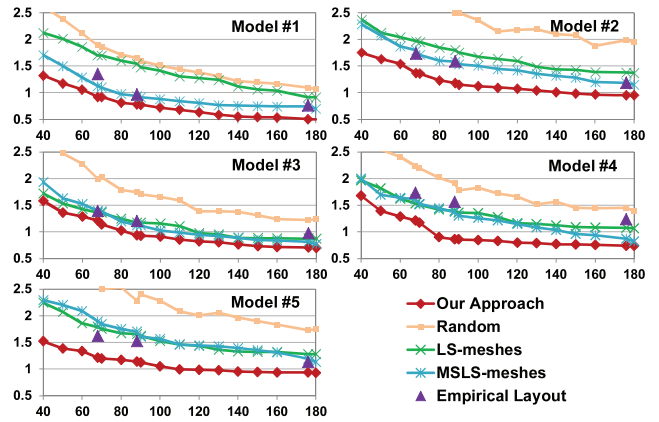


Fig. 12. RMS error curves of our approach, randomly distributed layouts, the two chosen baseline approaches (the LS-meshes approach [32] and the MSLS-meshes approach [33]), and the three different empirical facial marker layouts ([3], [4] and MPEG-4 facial animation). Numbers on the X -axes denote the numbers of markers. Numbers on the Y -axes denote the errors (unit: mm).

the MSLS-meshes on all the five data sets. As shown in this figure, our approach can consistently outperform those two approaches in terms of the vertex deformation error. Fig. 12 also plots the RMSE curves of our approach and all the baseline approaches on the five data sets. The errors for randomly distributed layouts and three empirical layouts are also plotted for comparison. As clearly shown in this figure, our approach can substantially outperform the baseline approaches for all the data sets, regardless the number of markers.

6.4 Cross Validations via Ground-Truth Data

We performed fivefold cross validations on all the five data sets. Basically, each data set (i.e., model #1 to #5) is first evenly partitioned into five subsequences (20 percent each). Then, for every data set, each of its five subsequences is retained as the test (validation) data while the other four subsequences are used as the training data. At the end, we average all the RMS errors to obtain the final RMS error for each model. Fig. 13 shows the obtained five fold cross-validation results of all the five data sets when varied numbers of markers are optimized. Again, the RMS errors of our method are clearly smaller than the others. We can also observe that the standard deviations of models #2 to #5 are clearly larger than that of model #1. The main reason is

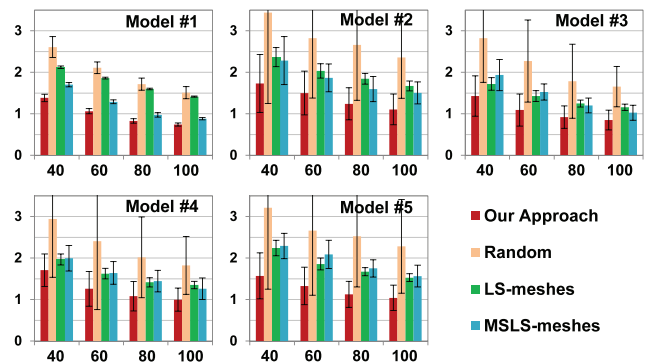


Fig. 13. Five fold cross-validation results on different data sets with varied numbers of optimized markers (X -axes). Numbers on the Y -axes denote the errors (unit: mm).

that facial expressions and movements are distributed, in a more balanced manner, in model #1 (from [8]) than the other four data sets (from [6]).

7 SELECTED APPLICATIONS

One primary application of our approach is to *guide minimal yet effective mocap marker placement for marker-based facial motion acquisition*. As shown in the cross-model validation results (Fig. 10), even if the optimized marker layouts are based on facial mesh sequence data of certain subjects, the layouts can still be soundly used for capturing the facial motions of another different subject. Furthermore, the computed facial mocap layouts shown in Fig. 8 can be used as practical facial marker placement guidelines in marker-based facial motion capture practices. As described in Section 6.3, the marker layouts optimized by our approach can consistently outperform selected empirical marker layouts and the marker layouts computed by the LS-meshes [32] and MSLS-meshes approaches [33] in terms of the average RMSE error.

Also, due to its mapping capability (i.e., transforming a deformed face mesh to the displacements of a set of characteristic control points), our approach can be potentially used for various applications including, but not limited to, facial animation editing, deformation, and compression. For the sake of a clear demonstration, in this section, we only detail two selected applications: *marker-based facial mesh skinning* (Section 7.1) and *multiresolution facial performance capture* (Section 7.2).

7.1 Marker-Based Facial Mesh Skinning

The facial marker layouts optimized by our approach can be directly used as a linear blend skinning method for animating facial mesh sequences, where the markers can be conceptually regarded as *proxy bones* [40]. The bone transformations influence every vertex on the facial mesh, subject to the weight matrix W (or called the bone-vertex influencing map). Compared with the general linear blend skinning, our marker-based facial skinning approach has the following three distinctive features. 1) The transformations of the proxy bones (i.e., the optimized markers) only contain the translation component but not rotation, scaling, or shearing components. 2) The weight matrix W can be computed solely from the rest pose given the determined (optimized) markers; thus, explicitly storing W at each frame is not needed. 3) Certain geometry properties (e.g., smoothness and fairness) of the skinned facial meshes can be soundly managed.

In the particular task of facial animation, the above first and second features support a more compact representation of the skinning model. Thus, it shall provide a better performance in certain applications such as compression or hardware accelerated rendering. Meanwhile, the above third feature provides an intuitive control over facial animation editing, where the markers can be used as draggable control points. By imposing desired geometry constraints (similar to how constraints can be imposed onto the thin-shell deformation model in this work), it would allow physically plausible editing on facial mesh sequences.

TABLE 4
Facial Skinning Comparison between Our Marker-Based Skinning, Our Marker-Based Skinning with Eigenskin Correction [41], and Key Point Subspace Acceleration [23]

Dataset	Parameters		RMSE _r (Compression Ratio)		
	# Mkrs	Rank	Ours	Ours + EC	KPSA
Model #1	40	20	1.598 _(106.8)	0.255 _(16.3)	0.771 _(17.1)
Model #1	60	30	1.256 _(98.1)	0.157 _(11.3)	0.656 _(11.7)
Model #1	80	40	1.080 _(90.6)	0.113 _(8.7)	0.626 _(8.9)
Model #1	100	50	0.865 _(84.2)	0.085 _(7.1)	0.502 _(7.2)
Model #2	40	20	1.985 _(102.7)	0.155 _(15.4)	0.976 _(16.1)
Model #2	60	30	1.560 _(95.1)	0.091 _(10.7)	0.609 _(11.1)
Model #2	80	40	1.306 _(88.5)	0.064 _(8.2)	0.459 _(8.4)
Model #2	100	50	1.170 _(82.8)	0.048 _(6.7)	0.369 _(6.8)
Model #3	40	20	2.062 _(25.8)	0.024 _(3.5)	0.087 _(3.6)
Model #3	60	30	1.440 _(25.3)	0.006 _(2.4)	0.014 _(2.5)
Model #3	80	40	1.264 _(24.9)	0.001 _(1.8)	0.004 _(1.9)
Model #3	100	50	0.943 _(24.4)	0.001 _(1.5)	0.001 _(1.5)
Model #4	40	20	2.472 _(68.7)	0.110 _(9.8)	0.365 _(10.3)
Model #4	60	30	1.723 _(65.2)	0.059 _(6.8)	0.262 _(7.0)
Model #4	80	40	1.162 _(62.0)	0.040 _(5.2)	0.155 _(5.4)
Model #4	100	50	0.912 _(59.2)	0.026 _(4.2)	0.128 _(4.3)
Model #5	40	20	1.720 _(92.9)	0.166 _(13.7)	0.999 _(14.4)
Model #5	60	30	1.474 _(86.6)	0.089 _(9.6)	0.709 _(9.9)
Model #5	80	40	1.254 _(81.1)	0.055 _(7.3)	0.704 _(7.5)
Model #5	100	50	1.172 _(76.3)	0.038 _(5.9)	0.243 _(6.1)

The root mean squared reconstruction error is reported for all the methods with the same parameters, i.e., the same number of control points (markers) and the same rank (the number of basic vectors). The corresponding compression ratio is also reported in the parentheses.

The purpose of our marker-based facial skinning shares certain similarities with the Key Point Subspace Acceleration (KPSA) technique [23]. However, while our marker-based facial mesh skinning essentially relies on the linear blend skinning method, KPSA uses the linear subspace deformation model. For this reason, KPSA requires an additional step of building a subspace from a set of facial pose examples. In addition, the animation control on example poses is limited to this subspace, avoiding crafting any facial expression that is not a linear combination of poses in the subspace. In term of compactness, KPSA also requires more data to represent the set of example poses (i.e., a full set of basis vectors), compared with a small amount of data requirement in our approach (i.e., the rest pose and the optimized marker layout).

In Table 4, we compare the RMSE reconstruction error between our marker-based facial skinning approach and the KPSA approach. For a fair comparison, we do not impose the symmetry and boundary constraints in our approach. Also, the number of markers (in our approach) and the number of control points (in the KPSA approach) are enforced to be equal. Note that the number of control points in KPSA is two times the number of basis vectors. From the comparison results, we can see that the RMSE errors of our method (column 4) are higher than those of KPSA (column 6). Clearly, there is a tradeoff between compactness and the RMSE error in our marker-based facial skinning method. However, we can control this tradeoff by applying skinning correction methods such as the EigenSkin correction (EC) [41]. In Table 4, the rank of the EigenSkin correction is set to be the same as the number of basic

vectors in KPSA for the sake of a fair comparison, i.e., a half of the number of markers. Since our method does not require to store the weight matrix, W , the amount of data in the compact form for both the methods, 1) KPSA and 2) marker-based facial skinning + EigenSkin correction, will be approximately the same, i.e., equal to the rank of the EigenSkin correction (or the number of basic vectors in KPSA) multiplied by the number of mesh vertices. Our marker-based facial skinning + EigenSkin correction (column 5) can significantly outperform the KPSA method (column 6), as shown in Table 4.

7.2 Multiresolution Facial Performance Capture

As mentioned in Section 2, in optical motion capture systems there is a tradeoff between robustness and capture resolution. Marker-based capture techniques, in general, offer more robust tracking at a higher frame rate, while markerless performance capture techniques can capture minute surface details in a higher resolution but at a lower frame rate. Researchers have proposed various hybrid (or called multiresolution) capture solutions [10], [11] to have the complementary advantages of the two methods. We believe such multiresolution performance capture techniques will benefit from our proposed marker optimization work. For example, the result by multiresolution capture techniques would be more accurate if the large scale motion (i.e., motion of the markers) can be recorded via the optimized marker layouts suggested by our approach. Note that even our method requires a certain amount of facial performance capture data as the training input, it is not a chicken-and-egg problem for multiresolution performance capture techniques, because we can always first analyze a short sequence of facial performance capture data to optimize the marker layout and then use it for next captures.

8 DISCUSSION AND CONCLUSIONS

In this paper, we propose a quantitative approach to optimize marker layouts for marker-based facial mocap and deformation. Specifically, by using several acquired high-resolution facial mesh sequences as the ground-truth data sets, we formulate this problem as a linear constrained optimization, where the reconstruction errors of a set of facial example poses are minimized with respect to a set of markers. The thin-shell linear deformation model is chosen to optionally impose hard constraints (including boundary constraint, symmetry constraint, and multiresolution constraint) onto the mesh reconstruction process to tailor the results toward different applications. Through various validations and comparisons, we demonstrate the accuracy, robustness, and usefulness of our approach.

Our method can be directly used for facial motion acquisition and deformation such as guiding the design of a minimal yet effective facial mocap marker set—a long-standing problem remaining to be resolved in marker-based facial motion capture. In addition, we also describe and demonstrate the effectiveness of using our approach for marker-based facial mesh skinning and multiresolution facial performance capture applications. Also, due to its intrinsic transformation capability (i.e., transforming a deformed face mesh to the displacements of a set of

characteristic control points), our approach can be potentially used for many other applications such as facial animation compression, deformation, and editing.

However, there are several limitations in the current approach, as described below:

- First, like numerous other data driven approaches, the marker layouts optimized by our approach depend on the nature of the training facial mesh sequence data. If the training facial mesh sequence data lack a variety of facial expressions and movements, the optimized marker layouts by our approach may not be perfectly balanced and distributed throughout the face.
- Second, the employed linear deformation model cannot faithfully capture the subtle deformation of the human face. By employing a nonlinear skin correction method (e.g., EigenSkin [41]), we are able to bring down the reconstruction error. However, this cannot be seamlessly integrated to our current framework.
- Third, because our formulated problem is a discrete, multivariable optimization, finding the global optimum is challenging. Although our marker refinement algorithm can decrease the error in our experiments, a monotonic descending solution with mathematical proof is still beyond the reach.
- Finally, our optimized deformation error in the least-squares sense may not always guarantee the best visual result. From the perceptual perspective, distortion in certain facial regions (e.g., the mouth and the eye/eyebrow regions) could be more perceptually salient than the other regions; thus, these salient regions should be treated with higher importances during the marker optimization process. This perceptual saliency can be incorporated into our framework by employing the weighted linear least squares to solve the expected marker trajectory (Section 4.4).

In the future, we plan to take a deep look into several aspects of the problem to overcome the above limitations. For example, we can enhance the training data by designing balanced training data sets. Gender, age, and ethnicity could be additional factors to be taken into consideration. The limitation of the linear deformation model can be impinged by combining subspace deformation models that have been widely used for facial animation. Also, by properly absorbing perceptual insights from various psychological and psychophysical facial experiments (e.g., perceptual saliency can be incorporated as an additional weight per mesh vertex), we could further improve the soundness and accuracy of automated extraction of characteristic facial control points.

ACKNOWLEDGMENTS

This work was supported in part by US National Science Foundation (NSF) IIS-0914965 and generous research gifts from Google and Nokia. The authors would like to thank Li Zhang, Derek Bradley and Thabo Beeler for providing their 3D face data sets. They would also like to deeply thank Hao (Richard) Zhang at the Simon Fraser University for his numerous helps and insightful discussion on this work. Any

opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the agencies. This work was done while Mingyang Zhu conducted his visiting research at the University of Houston from 09/2010 to 08/2012.

REFERENCES

- [1] G. Welch and E. Foxlin, "Motion Tracking: No Silver Bullet, but a Respectable Arsenal," *IEEE Computer Graphics and Applications*, vol. 22, no. 6, pp. 24-38, Nov./Dec. 2002.
- [2] D. Vlastic, R. Adelsberger, G. Vannucci, J. Barnwell, M. Gross, W. Matusik, and J. Popović, "Practical Motion Capture in Everyday Surroundings," *ACM Trans. Graphics*, vol. 26, article 35, July 2007.
- [3] B. Bickel, M. Lang, M. Botsch, M.A. Otaduy, and M. Gross, "Pose-Space Animation and Transfer of Facial Details," *Proc. ACM SIGGRAPH Eurographics Symp. Computer Animation (SCA '08)*, pp. 57-66, 2008.
- [4] W.-C. Ma, A. Jones, J.-Y. Chiang, T. Hawkins, S. Frederiksen, P. Peers, M. Vukovic, M. Ouhyoung, and P. Debevec, "Facial Performance Synthesis Using Deformation-Driven Polynomial Displacement Maps," *ACM Trans. Graphics*, vol. 27, pp. 121:1-121:10, Dec. 2008.
- [5] H. Woltring, "New Possibilities for Human Motion Studies by Real-Time Light Spot Position Measurement," *Biotelemetry*, vol. 1, pp. 132-146, 1974.
- [6] D. Bradley, W. Heidrich, T. Popa, and A. Sheffer, "High Resolution Passive Facial Performance Capture," *ACM Trans. Graphics*, vol. 29, pp. 41:1-41:10, July 2010.
- [7] T. Beeler, F. Hahn, D. Bradley, B. Bickel, P. Beardsley, C. Gotsman, R.W. Sumner, and M. Gross, "High-Quality Passive Facial Performance Capture Using Anchor Frames," *ACM Trans. Graphics*, vol. 30, pp. 75:1-75:10, Aug. 2011.
- [8] L. Zhang, N. Snavely, B. Curless, and S.M. Seitz, "Spacetime Faces: High Resolution Capture for Modeling and Animation," *Proc. ACM SIGGRAPH '04 Papers*, pp. 548-558, 2004.
- [9] E. de Aguiar, C. Theobalt, C. Stoll, and H.-P. Seidel, "Marker-Less Deformable Mesh Tracking for Human Shape and Motion Capture," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR '07)*, June 2007.
- [10] B. Bickel, M. Botsch, R. Angst, W. Matusik, M. Otaduy, H. Pfister, and M. Gross, "Multi-Scale Capture of Facial Geometry and Motion," *ACM Trans. Graphics*, vol. 26, no. 3, article 33, July 2007.
- [11] H. Huang, J. Chai, X. Tong, and H.-T. Wu, "Leveraging Motion Capture and 3D Scanning for High-Fidelity Facial Performance Acquisition," *Proc. ACM SIGGRAPH '11 Papers*, pp. 74:1-74:10, 2011.
- [12] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D.H. Salesin, "Synthesizing Realistic Facial Expressions from Photographs," *Proc. ACM SIGGRAPH '98*, pp. 75-84, 1998.
- [13] T. Weyrich, W. Matusik, H. Pfister, B. Bickel, C. Donner, C. Tu, J. McAndless, J. Lee, A. Ngan, H.W. Jensen, and M. Gross, "Analysis of Human Faces Using a Measurement-Based Skin Reflectance Model," *ACM Trans. Graphics*, vol. 25, no. 3, pp. 1013-1024, July 2006.
- [14] Z. Deng and J.Y. Noh, "Computer Facial Animation: A Survey," *Data-Driven 3D Facial Animation*, pp. 1-28, Springer, Nov. 2007.
- [15] J.-y. Noh and U. Neumann, "Expression Cloning," *Proc. ACM SIGGRAPH '01*, pp. 277-288, 2001.
- [16] R.W. Sumner and J. Popović, "Deformation Transfer for Triangle Meshes," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 399-405, Aug. 2004.
- [17] E. Sifakis, I. Neverov, and R. Fedkiw, "Automatic Determination of Facial Muscle Activations from Sparse Motion Capture Marker Data," *Proc. ACM SIGGRAPH '05 Papers*, pp. 417-425, 2005.
- [18] T. Weise, H. Li, L. van Gool, and M. Pauly, "Face/Off: Live Facial Puppetry," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA '09)*, pp. 7-16, 2009.
- [19] T. Weise, S. Bouaziz, H. Li, and M. Pauly, "Realtime Performance-Based Facial Animation," *ACM Trans. Graphics*, vol. 30, no. 4, pp. 77:1-77:10, Aug. 2011.
- [20] Z. Deng, P.-Y. Chiang, P. Fox, and U. Neumann, "Animating Blendshape Faces by Cross-Mapping Motion Capture Data," *Proc. Symp. Interactive 3D Graphics and Games (I3D '06)*, pp. 43-48, 2006.
- [21] Y. Seol, J.P. Lewis, J. Seo, B. Choi, K. Anjyo, and J. Noh, "Spacetime Expression Cloning for Blendshapes," *ACM Trans. Graphics*, vol. 30, no. 6, pp. 14:1-14:12, 2011.
- [22] P. Joshi, W.C. Tien, M. Desbrun, and F. Pighin, "Learning Controls for Blend Shape Based Realistic Facial Animation," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA '03)*, pp. 187-192, 2003.
- [23] M. Meyer and J. Anderson, "Key Point Subspace Acceleration and Soft Caching," *ACM Trans. Graphics*, vol. 26, article 74, July 2007.
- [24] W.-W. Feng, B.-U. Kim, and Y. Yu, "Real-Time Data Driven Deformation Using Kernel Canonical Correlation Analysis," *ACM Trans. Graphics*, vol. 27, pp. 91:1-91:9, Aug. 2008.
- [25] H. Li, T. Weise, and M. Pauly, "Example-Based Facial Rigging," *ACM Trans. Graphics*, vol. 29, no. 4, pp. 32:1-32:6, July 2010.
- [26] J.R. Tena, F. De la Torre, and I. Matthews, "Interactive Region-Based Linear 3D Face Models," *Proc. ACM SIGGRAPH '11 Papers*, pp. 76:1-76:10, 2011.
- [27] C. Bregler, M. Covell, and M. Slaney, "Video Rewrite: Driving Visual Speech with Audio," *Proc. ACM SIGGRAPH '97*, pp. 353-360, 1997.
- [28] M. Brand, "Voice Puppetry," *Proc. ACM SIGGRAPH '99*, pp. 21-28, 1999.
- [29] T. Ezzat, G. Geiger, and T. Poggio, "Trainable Videorealistic Speech Animation," *ACM Trans. Graphics*, vol. 21, no. 3, pp. 388-398, July 2002.
- [30] Z. Deng and U. Neumann, "eFASE: Expressive Facial Animation Synthesis and Editing with Phoneme-Isomap Controls," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA '06)*, pp. 251-260, 2006.
- [31] K. Wampler, D. Sasaki, L. Zhang, and Z. Popović, "Dynamic, Expressive Speech Animation from a Single Mesh," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA '07)*, pp. 53-62, 2007.
- [32] O. Sorkine and D. Cohen-Or, "Least-Squares Meshes," *Proc. Shape Modeling Int'l (SMI '04)*, pp. 191-199, 2004.
- [33] R. Southern and J.J. Zhang, "Motion-Sensitive Anchor Identification of Least-Squares Meshes from Examples," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 6, pp. 850-856, June 2011.
- [34] M. Garland and P.S. Heckbert, "Surface Simplification Using Quadric Error Metrics," *Proc. ACM SIGGRAPH '97*, pp. 209-216, 1997.
- [35] M. Botsch and O. Sorkine, "On Linear Variational Surface Deformation Methods," *IEEE Trans. Visualization and Computer Graphics*, vol. 14, no. 1, pp. 213-230, Jan. 2008.
- [36] M. Meyer, M. Desbrun, P. Schröder, and A. Barr, "Discrete Differential Geometry Operators for Triangulated 2-Manifolds," *Proc. Int'l Workshop Visualization and Math.*, citeseer.ist.psu.edu/meyer02discrete.html. 2002.
- [37] D.P. Bertsekas, *Nonlinear Programming*, second ed. Athena Scientific, Sept. 1999.
- [38] T.A. Davis and W.W. Hager, "Dynamic Supernodes in Sparse Cholesky Update/Downdate and Triangular Solves," *ACM Trans. Math. Software*, vol. 35, no. 4, pp. 27:1-27:23, Feb. 2009.
- [39] Z. Mao, X. Ju, J.P. Siebert, W.P. Cockshott, and A. Ayoub, "Constructing Dense Correspondences for the Analysis of 3D Facial Morphology," *Pattern Recognition Letters*, vol. 27, no. 6, pp. 597-608, Apr. 2006.
- [40] J. Gain and D. Bechmann, "A Survey of Spatial Deformation from a User-Centered Perspective," *ACM Trans. Graphics*, vol. 27, pp. 107:1-107:21, no. 4, Nov. 2008.
- [41] P.G. Kry, D.L. James, and D.K. Pai, "EigenSkin: Real Time Large Deformation Character Skinning in Hardware," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA '02)*, pp. 153-159, 2002.



Binh H. Le received the BS degree in computer science from the Vietnam National University, Vietnam, in 2008, and is currently working toward the PhD degree at the Department of Computer Science, University of Houston (UH) under the supervision of Prof. Zhigang Deng. His research interests include computer graphics, computer animation, and virtual human modeling and animation. He held a Vietnam Educational Foundation Fellowship from 2008 to 2010.



the University of Houston under the supervision of Prof. Zhigang Deng.

Mingyang Zhu received both the BS and MS degrees in computer science from the Nanjing University of Science and Technology in 2003 and 2007, respectively, where he is currently working toward the PhD degree at the Department of Computer Science. His research interests include computer graphics and character animation. From September 2010 to August 2012, he was a visiting PhD student at the Computer Graphics and Interactive Media Lab at



(CGIM) Lab. His research interests include computer graphics, computer animation, virtual human modeling and animation, and human computer interaction. He is a senior member of the IEEE, a member of the ACM, and a Board member of the International Chinese Association of Human Computer Interaction.

Zhigang Deng received the BS degree in mathematics from Xiamen University, China, the MS degree in computer science from Peking University, China, and the PhD degree in computer science from the Department of Computer Science, University of Southern California in 2006. He is currently an associate professor of computer science at the University of Houston (UH) and the founding director of the UH Computer Graphics and Interactive Media

▷ **For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.**