COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2011; **22**:159–167
Published online 12 April 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.404

SPECIAL ISSUE PAPER

A framework for locally retargeting and rendering facial performance

Ko-Yun Liu[1,2]*, Wan-Chun Ma[3]**, Chun-Fa Chang[4]***, Chuan-Chang Wang[1] and Paul Debevec[3]

[1] Next Media Animation, Taiwan
[2] Department of Computer Science, National Tsing-Hua University, Taiwan
[3] Institute of Creative Technologies, University of Southern California, CA, USA
[4] Department of Computer Science and Information Engineering, National Taiwan Normal University, Taiwan

ABSTRACT

We present a facial motion retargeting method that enables the control of a blendshape rig according to marker-based motion capture data. The main purpose of the proposed technique is to allow a blendshape rig to create facial expressions, which conforms best to the current motion capture input, regardless the underlying blendshape poses. In other words, even though all of the blendshape poses may comprise symmetrical facial expressions only, our method is still able to create asymmetrical expressions without physically splitting any of them into more local blendshape poses. An automatic segmentation technique based on the analysis of facial motion is introduced to create facial regions for local retargeting. We also show that it is possible to blend normal maps for rendering in the same framework. Rendering with the blended normal map significantly improves surface appearance and details. Copyright © 2011 John Wiley & Sons, Ltd.

KEYWORDS
expression synthesis; facial retargeting; normal map blending

***Correspondence**
Ko-Yun Liu, Next Media Animation, Taiwan.
E-mail: coco.kyliu@gmail.com

****Correspondence**
Wan-Chun Ma, Institute of Creative Technologies, USC.
E-mail: ma@ict.usc.edu

*****Correspondence**
Chun-Fa Chang, National Taiwan Normal University.
E-mail: chunfa@ntnu.edu.tw

1. INTRODUCTION

Marker-based facial motion capture is one of the favored implementations for performance driven facial animation. It often works together with blendshapes to create realistic facial animation. There are several reasons to associate motion capture with a blendshape rig as follows:

(1) The number of motion capture markers on an actor's face is considerably limited. A mesh built from even a few hundred motion capture markers will be in very low resolution such that we can only recognize major expressions from it. Merely using the mesh for animation is impossible to demonstrate all the details that are required to represent a realistic face. Therefore, motion capture data will usually be retargeted to a higher resolution geometric model in practice.
(2) It is hard for an animator to control blendshapes once there are too many blendshape poses. A solution is to have a high-level control user interface that imitates most of the possible facial movements, as shown in Reference [1]. However, building a facial rig with such a feature is also a nontrivial process.
(3) There are demands to incorporate motion capture data. For example, driving an animation with motion capture data significantly economizes labors and time comparing to tuning blendshape weights manually according to the reference images (rotoscoping). Besides, motion capture is able to acquire subtle expressions.

The key process is to convert the positions of facial motion capture markers into weights that control blendshapes, a process called *motion retargeting* [2]. A common technique for building a blendshape rig is to craft blendshape poses directly from real human facial 3D scans in various expressions. These expressions can be based on emotions, such as joy, sadness, anger, and so on; or they can be more comprehensively based on the Facial Action Coding System (FACS) [3], where every expression is motivated by muscle group activation. A well-trained digital modeler will then begin to build face shapes with consistent texture coordinate (correspondence) based on the scans. However, usually these scans were captured with combinational or symmetrical facial expressions. For example, we will capture the *eyes-closed* expression with both the subject's eyes closed in one single scan. This is essential in order to get through the scanning session quicker and to save data processing time. However, blendshape poses built from those scans have to be split up into several localized shapes to provide the full amount of control over each region of the face. Taking the *eyes-closed* again for example, the shape will probably be split up into two shapes: *left-eye-closed* and *right-eye-closed*, to allow controlling the two eyes independently. However, the splitting mechanism inevitably leads to a large number of blendshape poses. Eventually it becomes an issue to have an intuitive interface for controlling the rig. We propose a technique that is able to accommodate *localized* control without splitting the blendshape poses. Motion capture data will be the only source to drive the blendshape rig.

Our technique automatically generates segmentation masks which divide the face into many local regions. We optimize the best blendshape weights to allow independent control over each local region. All the vertices in a local region share the same blendshape weights. If a vertex is covered by more than one mask, blending is required based on the masks.

The contributions of our work are as follows:

(1) A straightforward technique to locally retarget motion capture data to a blendshape rig. Our method is able to produce arbitrary facial expressions which conform better to the motion capture data input regardless of the underlying blendshape poses. Our method introduces a blending (scaling) matrix into the retargeting computation, thereby predicting better blendshape weights for each local region.
(2) An automatic segmentation technique which enables local retargeting of a blendshape rig. It first analyzes the variation of each vertex in a blendshape rig by looking into the principal direction that a vertex tends to move along with, then clusters vertices with similar directions.
(3) The proposed normal map blending technique is able to create realistic facial animation in real-time. Our results demonstrate significantly visual quality improvements over traditional static normal mapping, and potentially can be applied in real-time rendering applications such as video games.

2. RELATED WORK

Lance Williams [4] introduced performance driven facial animation that tracked real facial motion from the actors and drove other 3D face models accordingly. Since then, there have been many studies on how to transfer a real performance to a controllable rig, especially in the scope of using marker-based motion capture data. A good survey of facial motion retargeting can be found in Reference [2].

2.1. Mocap-driven Facial Animation

The most relevant literature to our work is Reference [5]. In that work, an automatic segmentation technique is proposed to divide the face into regions that have similar amount of deformation. Each region finds its best blendshape weights that conforms to the motion capture data and then propagates the weights to each vertex by radial basis functions (RBFs) [6]. In our case, regions are clustered based on motions. Besides, since it is not easy to find the precise mapping between motion capture markers and their the corresponding locations in blendshapes, we establish a corresponding set of motion capture data for each of the blendshape poses. As a result, we only need a training session that records motion capture data that matches a predefined set of facial expressions for direct retargeting.

Chuang [7] presented a retargeting framework for creating facial animation using a weighted combination of the example face shapes from decomposed input video. Retargeting is carried out by creating corresponding target blendshape poses for an animated face model.

Deng *et al.* [8] solved the retargeting problem by using scattered data interpolation. The dimensionality of the motion capture data is first reduced using principal component analysis (PCA), so that each motion capture data can be projected onto a small set of parameters. The PCA weights of a given input motion capture frame are then interpolated by RBFs and manually adjust the blendshape rig to match images of reference poses for retargeting. In addition, Pyun *et al.* [9] and Li *et al.* [10] use PCA for motion retargeting. One major disadvantage of using PCA is that the blendshape weights derived from PCA are very poorly suited for manual editing, so the application must be fully automatic. To benefit from blendshape control in the post-production session, our algorithm chooses to directly manipulate the blendshapes in their original space instead of using a PCA subspace.

Buck *et al.* [11] proposed an algorithm for retargeting facial motion to hand-drawn animation. The technique showed that it is possible to render captured facial performance data by blending a small number of hand-drawn cartoons of characters in different facial expressions. Based on Reference [11], Hawkins *et al.* [12] presented a technique for creating an animatable image-based appearance model of a human face by capturing appearance over changing facial expressions. Their result shows that the subject's face can be rendered realistically under any linear combination of the input face shapes and under any desired lighting condition by using the the original images for relighting.

Liu *et al*. [13] presented a method to compute blendshape weights from facial motion capture data. Instead of using PCA, they used the *k*-means cluster method to cluster facial motion capture data, and then select those frames which are nearest to each cluster center to construct the subset of motion. However, Liu *et al*. [13] did not discussed how to locally control blendshapes.

Bickel *et al*. [14] and [15] used no blendshapes, but a fast linear shell model to deform a low resolution 3D face model through motion capture data. Fine-scale facial details are then synthesized by either using shape from shading [14] or pose–space deformation [15]. Ma *et al*. [16] proposed a technique to infer multi-resolution facial geometry using motion capture data. To analyze the correlation between the lowest resolution deformation (motion capture data) and higher resolution deformation (3D facial geometry and displacement maps), they acquired the subject's motion capture data and scan his face in 3D while performing a set of training expressions at the same time. The results showed that it is possible to create a detailed animatable facial geometry solely by a sparse set of motion capture markers. However, their method requires a complicated lighting system and a real-time 3D scanning system, which is not easily accessible. Retargeting is also nontrivial because facial deformation differs from person to person. Similarly, Sanchez [17] estimated surface normal maps (instead of displacement maps) in a similar fashion.

2.2. Other Facial Animation Techniques

Guenter *et al.* [18] designed a multi-camera system for capturing both the geometry and texture map of a facial performance. The multi-view video data is then assembled in 3D to create realistic animations of the captured expressions.

Similar to Reference [18], the Universal Capture system [19] used a multi-camera system to capture facial performance. However, instead of using facial markers for tracking, they used optical flow to track motion of each pixel over time in each camera view. Each vertex in a 3D model of a neutral expression of the actor is adjusted according to the optical flow data. The 2D projected position of a vertex in each view is first determined, then its 3D position is estimated using epipolar geometry. The result is an accurate reconstruction of the motion path of each vertex over time.

Alexander *et al.* [1] demonstrated a video-driven facial animation technique in the Digital Emily Project. However, their animation pipeline is only semi-automatic. It requires an experienced digital artist to adjust the blendshape rig for certain key-frames so that the results match best to the input video, then the manually edited blendshape weights can be provided as initial guesses to the retargeting algorithm.

3. FACIAL MOTION RETARGETING

3.1. Global Retargeting

To drive the blendshape model, we first create a corresponding set of motion capture data for each of the blendshape poses. It is carried out by capturing key poses, which bijectively map to the blendshape poses, from an actor. Since the blendshape weights will be computed solely on motion capture data, the motion capture actor may not necessarily be the same person whom we scan for the digital actor as long as a good bijective mapping is established.

We define a set of motion capture basis vectors $\mathbf{s}_i$ and a target vector $\mathbf{t}$ as

$$\mathbf{s}_i = [\mathbf{x}(p_1^{s_i}), \mathbf{y}(p_1^{s_i}), \mathbf{z}(p_1^{s_i}), \cdots, \mathbf{x}(p_{n_\mathrm{m}}^{s_i}), \mathbf{y}(p_{n_\mathrm{m}}^{s_i}), \mathbf{z}(p_{n_\mathrm{m}}^{s_i})]^T,$$
$$\mathbf{t} = [\mathbf{x}(p_1^{t}), \mathbf{y}(p_1^{t}), \mathbf{z}(p_1^{t}), \cdots, \mathbf{x}(p_{n_\mathrm{m}}^{t}), \mathbf{y}(p_{n_\mathrm{m}}^{t}), \mathbf{z}(p_{n_\mathrm{m}}^{t})]^T.$$

n_m is the number of markers. $\mathbf{x}(p_i)$, $\mathbf{y}(p_i)$, and $\mathbf{z}(p_i)$ are functions that return the X, Y and Z Cartesian coordinates of a 3D position p_i. We can put all the basis vectors into a single matrix $\mathbf{S} = [s_1, s_2, \ldots, s_{n_\mathrm{b}}]$, where n_b is the number of blendshape poses. The recovery of blendshape weights w for a given shape t can be posed into a constrained least squares problem

$$\underset{1\geq \mathbf{w}(i)\geq 0,\quad \sum_{i=1}^{n_\mathrm{b}} \mathbf{w}(i)=1}{\arg\min} \|\mathbf{S}\mathbf{w}-\mathbf{t}\|^2. \qquad (1)$$

As described in Reference [20], applying the same weight vector $\mathbf{w}$ to the blendshape model yields a face shape which conforms the best to the input motion capture data $\mathbf{t}$. We define $\bar{s}_i$ to be a blendshape pose that corresponds to a motion capture basis $\mathbf{s}_i$:

$$\bar{s}_i = [\mathbf{x}(p_1^{\bar{s}_i}), \mathbf{y}(p_1^{\bar{s}_i}), \mathbf{z}(p_1^{\bar{s}_i}) \cdots, \mathbf{x}(p_{n_\mathrm{v}}^{\bar{s}_i}), \mathbf{y}(p_{n_\mathrm{v}}^{\bar{s}_i}), \mathbf{z}(p_{n_\mathrm{v}}^{\bar{s}_i})]^T.$$

n_v is the number of vertices in a blendshape pose. Similar to $\mathbf{S}$, $\overline{S} = [\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_{n_\mathrm{b}}]$, and the resulting face shape $\bar{t}$ can be computed as $\bar{t} = \overline{S}w$.

3.2. Local Retargeting

A large number of our blendshape shapes have symmetrical facial expressions, by just linearly combining the shapes will not produce asymmetrical facial expressions. Usually a digital artist will split such symmetrical shape into more localized shapes. However, it is inevitably a time-consuming process.

3.2.1. Automatic Segmentation.

To automatically create the segmentation masks, we propose an algorithm that makes use of the nature of facial action. Lewis and Anjyo [21] showed the relatedness of vertices in a human face. And we observe that a human face can be divided into a few regions such that each region is approximately independent of the others. For example, one

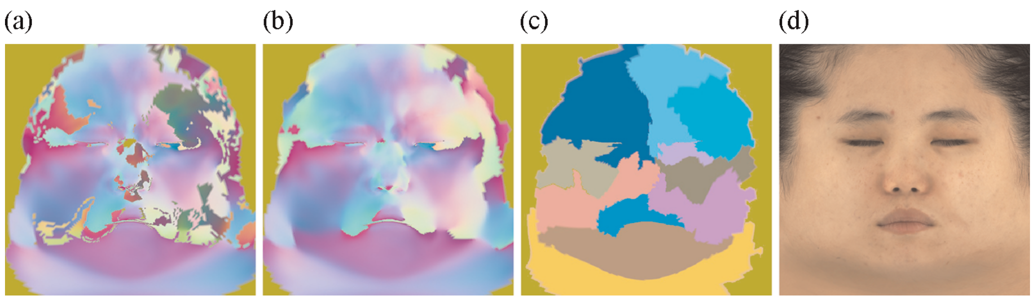

Figure 1. Automatic segmentation. (a) The directional field based on the facial motion PCA basis vector with the largest eigenvalue, shown in texture space. (b) Coordinated directional field by optimization. (c) Segmentation by K-Means clustering. (d) The texture space face image.

can move the eyebrows up and down without affecting the jaw much, or *vice versa*. For each vertex in the blendshape rig, we apply PCA on all its 3D positions from different shapes. The first principal component shows the direction that the vertex tends to move along and with the minimal summed squared error. This is the criterion we require to discover vertices that have similar motion so we can group them together. However, the principal components of adjacent vertices may point in opposite directions, as in Figure 1(a). Therefore, we have to coordinate the directions by a optimization process which assures the principal components of adjacent vertices do not differ excessively (i.e., for each vertex, minimizing the sum of dot products of the principal components between its neighbors). The optimization result is shown in Figure 1(b).

We then use k-means clustering algorithm to group blendshape vertices based on the coherent principal components. A cluster with fewer vertices than a preset threshold or with the fewest vertices is merged into its adjacent cluster which has the fewest vertices, and this process continues until the number of vertices of every mask is above a given threshold. A final segmentation is shown in Figure 1(c). Figure 1(d) shows a corresponding texture space face image for reference. We use the texture space triangulations of each group to create the masks. Each mask is then smoothed by a Gaussian filter in order to obtain a properly overlapping region for blending. These masks serve two purposes. First, they provide the scale factor for each motion capture marker that is used in the weighted least squares for motion retargeting. Second, they are also used to blend the blendshape geometry as shown in Section 4.

3.2.2. Retargeting.

We denote $\mathbf{M}_k$ as the scaling matrix derived from mask B_k

$$\mathbf{M}_k = \{m_{ij}^k | 1 \le i \le 3n_\mathrm{m}, 1 \le j \le 3n_\mathrm{m}\},$$
$$m_{ii}^k = \frac{\overline{m}_{ii}^k}{\sum_{k=1}^{n_\mathrm{r}} \overline{m}_{ii}^k},$$
$$\overline{m}_{ij}^k = \begin{cases} \mathbf{r}(B_k, q) & \text{if } i=j, q = \lfloor \frac{i-1}{3} \rfloor + 1 \\ 0 & \text{otherwise.} \end{cases}.$$

$\mathbf{M}_k$ is used to determine the importance of each motion capture marker in mask B_k (q is equivalent to the identification number of the motion capture marker). Function $\mathbf{r}(B_k, q)$ is implemented by averaging all the mask values within p_q's Vonoroi diagram cell in the texture space. n_r is the number of masks (or regions). If a motion capture marker is outside a given mask, we would assume that this marker has little or no influence compared to the markers which are covered by the mask. The following linear system is then constructed for solving blendshape weights for each region

$$\underset{1 \ge \mathbf{w}_k(i) \ge 0,\quad \sum_{i=1}^{n_\mathrm{b}} \mathbf{w}_k(i)=1}{\arg\min} \|\mathbf{M}_k\mathbf{S}\mathbf{w}_k - \mathbf{M}_k\mathbf{t}\|^2.$$

The above equation is able to be solved efficiently by quadratic programming [22]. It needs to be rewritten into the form of

$$\underset{A\mathbf{w}_k \ge b,\quad A_\mathrm{eq}\mathbf{w}_k = b_\mathrm{eq}}{\arg\min} \frac{1}{2}\mathbf{w}_k^T H \mathbf{w}_k + f^T \mathbf{w}_k, \qquad (2)$$

where $H = (\mathbf{M}_k\mathbf{S})^T(\mathbf{M}_k\mathbf{S})$, $f = -(\mathbf{M}_k\mathbf{t})(\mathbf{M}_k\mathbf{S})^T$, $A = [\mathrm{eye}(n_\mathrm{b}); -\mathrm{eye}(n_\mathrm{b})]$, $b = [\mathrm{ones}(1, n_\mathrm{b}), \mathrm{zeros}(1, n_\mathrm{b})]^T$, $A_\mathrm{eq} = [\mathrm{ones}(n_\mathrm{b}, 1)]$, $b_\mathrm{eq} = 1$.

Eye, zeros, and ones are functions that create a identity matrix, a matrix with all zeros, and a matrix with all ones, respectively. Since H is symmetric and positive definite, in this case there exists a global minimum if a feasible solution that satisfies the constraints can be found.

4. SYNTHESIZING NEW GEOMETRY

To create the face shape $\bar{t}$ that matches the current input motion capture data, we simply accumulate all the mask-modulated regional blendshape poses:

$$\bar{t} = \sum_{k=1}^{n_\mathrm{r}} B_k \bar{S} w_k. \qquad (3)$$

Figure 2 shows a comparison of globally and locally retargeted blendshape controls for a given expression. Local retargeting results conform better to the motion

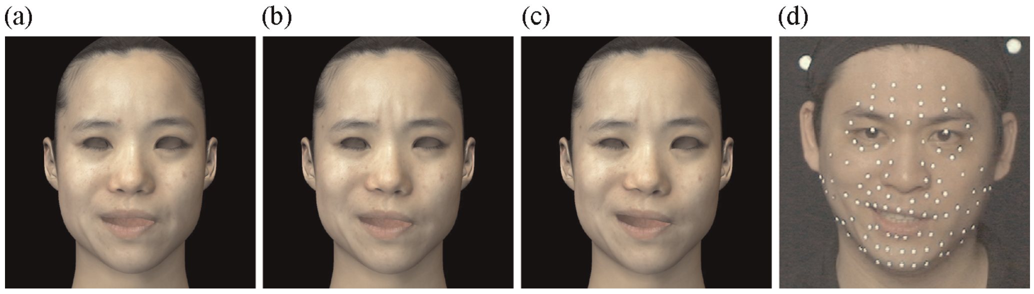

Figure 2. A comparison between global and local retargeting methods. (a) Global retargeting. (b) Local retargeting with six regions. (c) Local retargeting with 12 regions. (d) Reference photograph.

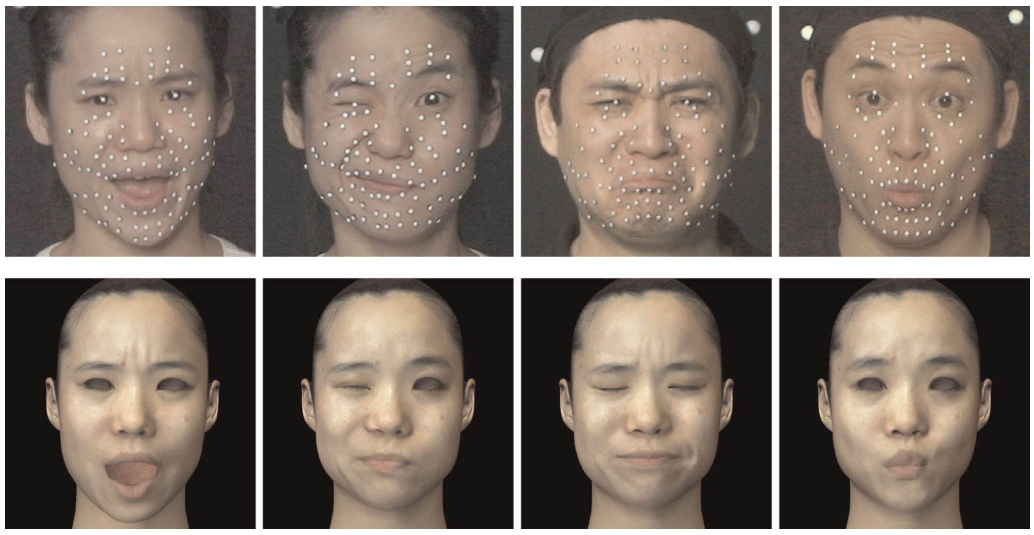

Figure 3. Retargeting facial motion from different motion capture actors. The top row shows reference photos of two different motion capture actors. The bottom row shows the blendshape rig with retargeted expressions.

capture input visually, and deliver smaller reconstruction errors (we will discuss error analysis later in Section 6). Figure 3 contains more results that are retargeted from two different motion capture actors.

5. NORMAL MAP BLENDING

Animated normal mapping has been demonstrated in Reference [17] for real-time blending (by pixel shader) of different normal maps, driven by local deformation. Ma *et al*. [16] used real-time scanned face data (with a total of seven expressions, each expression was captured from neutral to fully-activated). The geometry of each facial scan is stored in texture-space for blending. Both techniques used polynomials to fit the change of 3D coordinates or normal variation of a facial point according to a lower resolution deformation field, which is generated from the motion capture mesh.

Our scans come with a set of normal maps based on photometric stereo. Applying these accompanying normal maps can significantly improve surface appearance and details. It is intuitive to consider blending normal maps of different expressions under the same blendshape framework. Oat [23] shows it is possible to blend two different normal maps in a pixel shader. Similarly, we would like to actually blend normal maps according the corresponding blendshape weights and masks, in a way that is identical to Equation 3. However, in this case, the vertices stored in **S** in Equation 3 become normal vectors in texture space.

The only preprocessing that has to be done is to warp all the normal and displacement maps into a common texture space. The photometric stereo normal maps originally are in camera space. For each expression, we first align the original face scan with the blendshape shape, then for each vertex of the face scan we build the correspondence by searching for the closet vertex of the blendshape shape. We then use the correspondences to build a warping which is able to transfer textures from a face scan's texture space (a camera space) into blendshape rig's texture space.

To better correspond texture space maps, we compute optical flow between the two texture space displacement maps, one from a particular expression and the other from the neutral expression, as a correspondence for warping

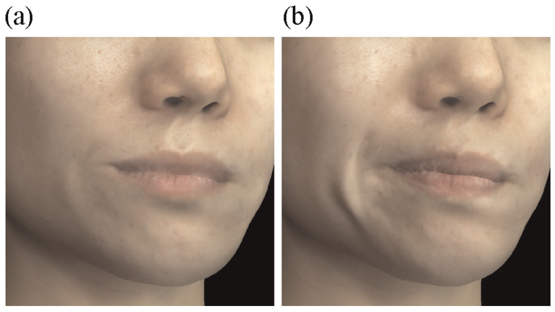

Figure 4. An example of normal map blending. (a) Rendering with only a static normal map from the neutral expression. (b) Rendering with blended normal maps is able to present more realistic surface details, such as the stretching of the lip wrinkles.

adjustment. The computed flow will provide better registration for the texture-space normal maps. It is important to have good alignment for all the the texture-space normals, otherwise fine features in the normal maps will be blurred. The same procedure could also be applied to albedo texture maps too, as an animated texture mapping technique. Figure 4 shows images which are rendered with or without blended normal maps for comparisons.

6. IMPLEMENTATION AND RESULTS

In this section, we describe some of the implementation aspects for the proposed local retargeting and rendering method. In addition, we show that our local retargeting method is able to produce less reconstruction error.

6.1. Building the Blendshape Rig

We have set up a facial capture system that is similar to Reference [24] to scan our character for blendshape reference. A set of 22 predefined different expressions were scanned. Four of these face scans are based on Preston Blair's phoneme series [25], one is the neutral expression, and another 17 scans are various basic expressions of human face. The subject being scanned is not necessarily the same person as the motion capture actor. Post-processing of the scans and building the corresponded blendshape poses were done manually by digital modelers with Autodesk Maya and Pixologic ZBrush software. The masks of regions around the two eyes are manually edited because we found that the vertices within the regions are few and the automatic segmentation technique inevitably merges those regions into other ones.

6.2. Performance

The key of the retargeting algorithm is solving Equation 2 with quadratic programming, which can be done in polynomial time. Our implementation is able to process 72.3 frames per second on a machine with a Intel Xeon X5550 2.67GHz CPU where the number of blendshape poses n_b is 22 and the number of markers n_m is 97.

6.3. Rendering

We incorporated the hybrid normal rendering technique in Reference [24] to achieve believable skin reflectance. We implemented a real-time renderer with Microsoft DirectX 9 and Shader Model 3.0 HLSL. The normal maps for the specular channel have 2048×2048 pixel resolution, a sufficient resolution such that most skin detail is preserved. The normal maps for diffuse channels (red, green, and blue), which contain fewer details, were stored at 1024×1024 pixel resolution. The renderer is capable of blending a total of 88 normal maps (each pose has four normal maps for the red, green, blue, and specular channels) on the fly, achieving 14.2 frames per second with a NVIDIA Quadro FX 4800 GPU. Our renderer also supports image-based lighting, which is approximated by prefiltered environment maps [26]. Figure 5 shows several screenshots from the final animation results.

6.4. Manual Post-editing

Our system incorporates directly with blendshapes and provide post-editing features. We have developed a tool for Maya that is capable of editing blendshape weight curves in one or multiple selected regions. Figure 6 shows results of an original motion retargeted rig and one after post-editing by an digital artist in order to provide a more visually rectified motion.

6.5. Error Analysis

As we expected, local retargeting improves upon global retargeting in the quality of reconstructed animation using blendshapes. Table 1 shows the comparison of reconstruction errors between global retargeting and local retargeting with various numbers of masks. The reconstruction error is measured as the average distance between each mocap marker and its reconstructed position during a typical facial performance. The average numbers show the trend of improvement and the maximum and minimum show that no marker behaves erratically when we use more facial regions for retargeting.

7. FUTURE WORK

There are several directions for future improvements. Currently the number of facial regions can not be explicitly determined. We have to set a proper threshold for the minimum number of vertices of each region, such that the

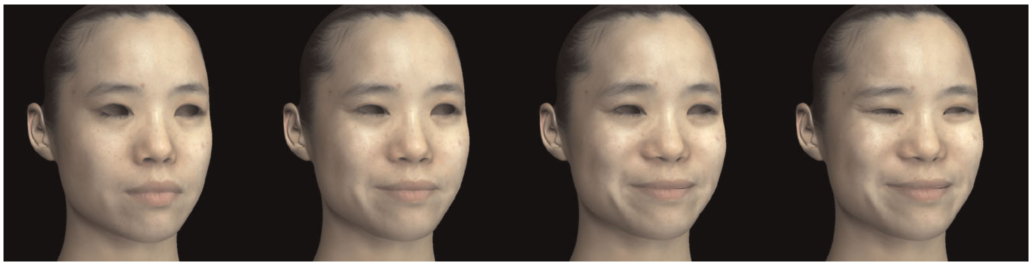

Figure 5. Screenshots from final results.

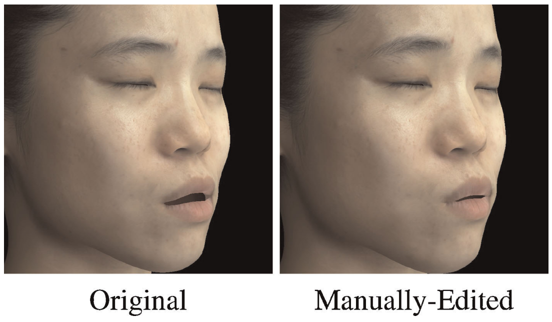

Figure 6. Post-editing blendshape weights. The blendshape weights in the lower face region was manually edited such that the rig visually conforms more to the digital artist's perspective.

Table 1. Average reconstruction errors (in mm) between global retargeting and local retargeting with various numbers of masks from a typical facial performance.

Group	Max.	Min.	Avg.	Vtx. thres.
Global	6.03	0.58	2.01	n/a
2	5.45	0.54	1.74	4000
6	4.41	0.42	1.44	1000
9	3.65	0.38	1.23	800
12	3.46	0.35	1.16	500

segmentation result matches a desirable number of facial regions. It would be preferred if we could set the number of regions directly and then the algorithm would split the blendshape rig accordingly.

Optimizing blendshape weights on a per-frame basis occasionally produces non-smooth animation curves. In the current system, to maintain a visually-pleasant motion, we smooth blendshape weights by applying a Gaussian filter in the temporal domain. We would instead like to integrate temporal smoothness equation such as in Reference [27] into our optimization process in the future.

ACKNOWLEDGEMENTS

The authors wish to thank the following people for their contributions to this paper: Jimmy Lai, Cassian Cheung, Ting-Chieh Lin, Tieh-Hung Chuang, Jackson Lee, Chian-Min Shih, and Bocheng Shih from Next Media Animation.

REFERENCES

1. Alexander O, Rogers M, Lambeth W, *et al*. The Digital Emily Project: achieving a photorealistic digital actor. *IEEE Computer Graphics and Applications* 2010; **30**(4): pages 20–31.
2. Pighin F, Lewis JP. Facial motion retargeting. In *SIGGRAPH Courses*, 2006.
3. Ekman P, Friesen W. *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists Press: Palo Alto, CA, 1978.
4. Williams L. Performance-driven facial animation. In *Proceedings of SIGGRAPH*, 1990; pages 235–242.
5. Joshi P, Tien WC, Desbrun M, Pighin F. Learning controls for blend shape based realistic facial animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003; pages 187–192.
6. Lewis JP, Cordner M, Fong. N. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of SIGGRAPH*, 2000; pages 165–172.
7. Chuang E. Analysis, synthesis, and retargeting of facial expressions. *Ph.D. Dissertation*, Stanford University, 2004.
8. Deng Z, Chiang P-Y, Fox P, Neumann U. Animating blendshape faces by cross-mapping motion capture data. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, 2006; pages 43–48.
9. Pyun H, Kim Y, Chae W, Kang HW, Shin. SY. An example-based approach for facial expression cloning. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003; pages 167–176.
10. Li Q, Deng. Z. Orthogonal-blendshape-based editing system for facial motion capture data. *IEEE Computer Graphics and Applications,* 2008; **28**(6): 76–82.
11. Buck I, Finkelstein A, Jacobs C, *et al*. Performance-driven hand-drawn animation. In *Proceedings of the International Symposium on Non Photorealistic Animation and Rendering*, 2000; pages 101–108.

12. Hawkins T, Wenger A, Tchou C, Gardner A, Göransson F, Debevec P. Animatable facial reflectance fields. In *Proceedings of the Eurographics Workshop on Rendering*, 2004; pages 309–320.
13. Liu X, Mao T, Xia S, Yu Y, Wang. Z. Facial animation by optimized blendshapes from motion capture data. *Computer Animation and Virtual Worlds* 2008; **19**(3-4): 235–245.
14. Bickel B, Botsch M, Angst R, *et al*. Multi-scale capture of facial geometry and motion. *ACM Transactions on Graphics* 2007; **26**(3): 33:1–33:10.
15. Bickel B, Lang M, Botsch M, Otaduy MA, Gross M. Pose-space animation and transfer of facial details. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2008; pages 57–66.
16. Ma W-C, Jones A, Chiang J-Y, *et al*. Facial performance synthesis using deformation-driven polynomial displacement maps. *ACM Transactions on Graphics* 2008; **27**(5): 121:1–121:10.
17. Sanchez M. Techniques for performance-based, real-time facial animation. *Ph.D. Dissertation*, University of Sheffield, 2006.
18. Guenter B, Grimm C, Wood D, Malvar H, Pighin F. Making faces. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 55–66, 1998.
19. Borshukov G, Piponi D, Larsen O, Lewis JP, Tempelaar-Lietz C. Universal capture: image-based facial animation for The Matrix Reloaded. In *SIGGRAPH Sketches*, 2003.
20. Pighin F, Hecker J, Lischinski D, Szeliski R, Salesin DH. Synthesizing realistic facial expressions from photographs. In *Proceedings of SIGGRAPH*, 1998; pages 75–84.
21. Lewis JP, Anjyo K. A region-of-influence measure for automatic skinning. In *Proceedings of Image and Vision Computing New Zealand*, 2007; pages 187–191.
22. Nocedal J, Wright SJ. *Numerical Optimization*, (2nd edn). Springer: New York, 2006.
23. Oat C. Animated wrinkle maps. In *SIGGRAPH Courses*, 2007.
24. Ma W-C, Hawkins T, Peers P, Chabert C-F, Weiss M, Debevec P. Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. In *Proceedings of the Eurographics Symposium on Rendering*, 2007; pages 183–194.
25. Martin GC. Preston Blair phomene series, 1997. http://www.garycmartin.com/mouth_shapes.html
26. Kautz J, Vázquez P-P, Heidrich W, Seidel H-P. A unified approach to prefiltered environment maps. In *Proceedings of the Eurographics Workshop on Rendering*, 2000; pages 185–196.
27. Lewis JP, Anjyo K. Direct manipulation blendshapes. *IEEE Computer Graphics and Applications* 2010; **30**(4): 42–50.

Authors' Biographies:

Ko-Yun Liu obtained the M.S. from National Tsing Hua University, Taiwan, in 2005. Currently, she is a Ph.D candidate at the Department of Computer Science, National Tsing Hua University, Taiwan, and works in a research and development team in Next Media Animation Ltd. Her research focuses on motion capture and facial animation.

Wan-Chun Ma is a postdoctoral research associate at the University of Southern California Institute for Creative Technologies. His research interests include 3D scanning, facial animation, motion capture, and general-purpose GPU techniques. Ma has a PhD in computer science from National Taiwan University. Contact him at ma@ict.usc.edu.

Chun-Fa Chang received the B.S. degree from National Taiwan University, Taipei, Taiwan in 1988, the M.S. from the Univeristy of Texas at Austin, U.S.A. in 1992, and the Ph.D. from the University of North Carolina at Chapel Hill, U.S.A. in 2001. His industrial experience includes a full-time position at Intel Corporation from 1992 to 1995, and summer internships at DEC-Western Research Lab in 1996 and 1997. He was an assistant professor in the Department of Computer Science, National Tsing Hua University from 2001 to 2007, and is currently an associate professor in the Department of Computer Science, National Taiwan Normal University. His research interests include real-time rendering, image-based modeling and rendering, photo-realistic image synthesis, and applications using multicore processors.

Chuan-Chang Wang holds a B.C. in Civil Engineering. Working in the computer graphics and game development industry for two decades, he developed real-time 3D rendering engine and set up a new production pipeline to improve the computer animation production from days to hours. He leads a research and development team in Next Media Animation Ltd. and works on the technologies about the creation of the digital actors. He is interested in real-time shader programming, computational geometry and global illumination rendering.

Paul Debevec is a research associate professor at the University of Southern California and the associate director of graphics research at USC's Institute for Creative Technologies. His work has focused on image-based modeling and rendering techniques beginning with his 1996 Ph.D. thesis at UC Berkeley, with specializations in architecture, high dynamic range lighting, and human facial capture. He serves on the ACM SIGGRAPH Executive Committee and recently received an Academy Award® for his work on the Light Stage facial capture systems.