Interactive 3D Facial Expression Posing through 2D Portrait Manipulation

Tanasai Sucontphunt*
Dept. of Computer Science
University of Houston

Zhenyao Mo †
University of Southern California

Ulrich Neumann‡
University of Southern California

Zhigang Deng §
Dept. of Computer Science
University of Houston

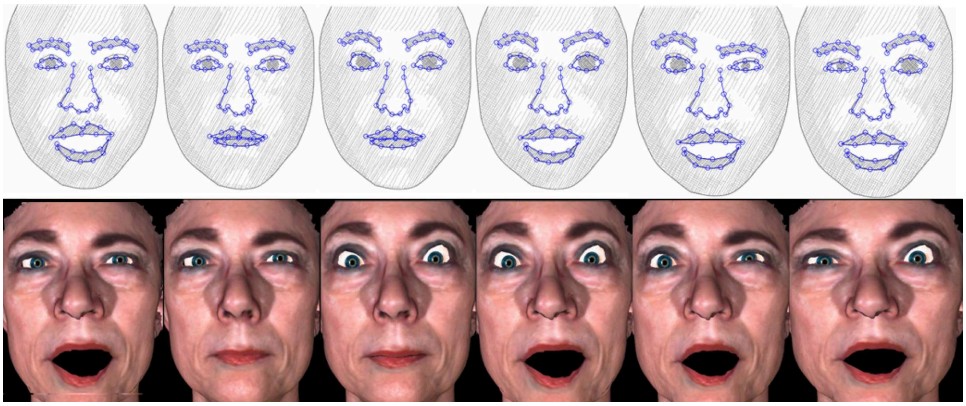

Figure 1: The top row shows 2D portraits edited by users. The bottom row shows generated 3D facial expressions.

ABSTRACT

Sculpting various 3D facial expressions from a static 3D face model is a process with intensive manual tuning efforts. In this paper, we present an interactive 3D facial expression posing system through 2D portrait manipulation, where a manipulated 2D portrait serves a metaphor for automatically inferring its corresponding 3D facial expression with fine details. Users either rapidly assemble a face portrait through a pre-designed portrait component library or intuitively modify an initial portrait. During the editing procedure, when the users move one or a group of 2D control points on the portrait, other portrait control points are adjusted in order to automatically maintain the faceness of the edited portrait if the automated propagation function (switch) is optionally turned on. Finally, the 2D portrait is used as a query input to search for and reconstruct its corresponding 3D facial expression from a pre-recorded facial motion capture database. We showed that this system is effective for rapid 3D facial expression sculpting through a comparative user study.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism— [H.5.2]: User Interfaces—Graphics User Interfaces (GUI)

1 INTRODUCTION

Sculpting realistic 3D facial expressions from a static 3D face model is a painstakingly manual process. Various research efforts including geometric deformation techniques and blendshape approaches have been attempted to ease the sculpting process. However, sculpting a sound blendshape face model required by the blendshape approaches [30, 21, 29] needs painstaking and intensive manual work, even for skilled animators; and it is difficult to sculpt or control the fine details of a deformed 3D face using geometric deformation techniques [23, 45, 41, 7] without considerable efforts.

Inspired by the work of [27, 19] that shows sketch-based interfaces are psychologically superior to images in identifying the human face features, we present an interactive system for rapidly posing 3D facial expressions with fine details through 2D portrait manipulation. This work exploits the fact that 2D portraits typically characterize prominent features of human faces [4], and editing portraits in a 2D space is more intuitive than directly working on 3D face meshes. While 2D portraits may not show certain face details, *e.g.*, wrinkles, this system automatically fills in these details of 3D face expressions in a data-driven way: based on an edited 2D portrait, it searches for the most matched 3D facial motion configurations in a pre-constructed facial motion database.

Another distinction of this system is that users can efficiently combine the characteristics of different facial configurations or even different human subjects to form a new facial expression that can be further edited. For example, assuming the users intend to combine the eyebrow/eye characteristics of an existing expression E_1 and the mouth characteristics of another existing expression E_2 to form a new facial expression E', traditional cut-and-paste methods or face sketching approaches [7, 26] that directly work in 2D image or 3D face model spaces would have hard time to accomplish this task. Furthermore, if the two expressive facial poses E_1 and E_2 are from two different individuals, then this task will become even more challenging. Using this system, users can efficiently accomplish the above task as follows: assemble a 2D portrait by picking proper portrait components (the eye/eyebrow components of E_1 and the mouth component of E_2) from a portrait component library and

*e-mail: tanasai@cs.uh.edu
†e-mail:zmo@usc.edu
‡email: uneumann@graphics.usc.edu
§e-mail:zdeng@cs.uh.edu

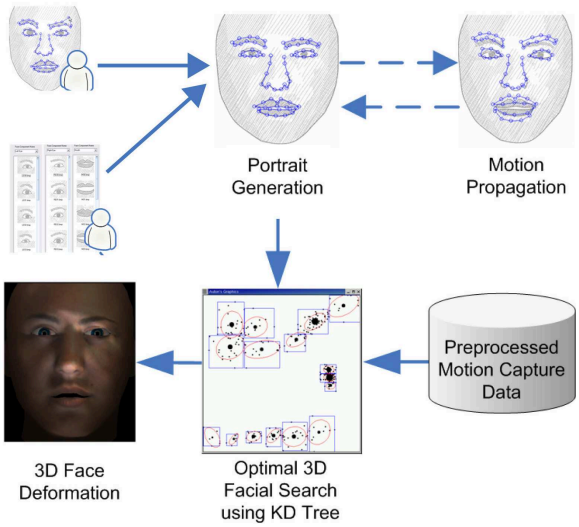

Figure 2: Schematic overview of this facial expression posing system. At runtime, users can either assemble a portrait from the portrait component library or edit an existing portrait. The edited portrait will be adjusted based on the movements of one or a group of control points. Finally, its corresponding 3D facial expression is inferred by searching through a KD-tree represented, pre-constructed facial motion database.

perform further 2D editing on it (if necessary), and then the expected new 3D facial expression E' will be accordingly sculpted.

In this system, users can either rapidly assemble a face portrait from a constructed portrait component library or edit a given portrait. When the users move one or a group of 2D control points of the portrait, the motion propagation algorithm [46] is optionally (users can turn this function on or off) used to automatically adjust other portrait control points to maximally maintain the faceness of the edited portrait. Finally, to map a 2D portrait to its corresponding 3D facial expression, the portrait is used as a query input to search for the most matched 3D facial configurations from a pre-constructed facial motion database, represented as a KD-tree. Figure 2 shows the schematic view of this 3D facial expression posing system.

The main contributions of this work include: (1) it introduces a new efficient tool for interactively posing 3D facial expressions, where a 2D portrait is used as an intuitive user interface and 3D facial details are automatically filled through a data-driven scheme, and (2) posing novel 3D facial expressions by combining prominent facial characteristics of existing facial expressions or different individuals provides a non-traditional while effective means to generating facial expressions and animations.

The remainder of this paper is organized as follows. A review of related work is given in Section 2. Section 3 describes facial motion data acquisition and preprocessing. Section 4 describes how to render 2D portraits using a model-based approach. Section 5 describes how to construct a portrait component library. Section 6 details how to perform editing operations on the 2D portraits. Section 7 describes how to infer or construct 3D facial expressions by searching for the most matched facial motion frames from a pre-recorded facial motion database, represented as a KD-tree. Experimental and usability study results are described in Section 8. Finally, in Section 9, concluding remarks are given and future research directions are discussed.

2 Related Work

Intensive facial animation research efforts have been pursued in the past several decades [36, 16]. Some of recent research efforts include 3D face modeling [28, 37, 1], speech animation synthesis [3, 2, 18, 15, 17] and facial animation transferring [35, 44, 40]. In this section we briefly review recent research attempts that are most related to this work.

Creating various facial expressions by manipulating thin-shell meshes (3D face models) is a popularized method in industry practice [12, 23, 43, 45, 41]. For example, the Free Form Deformation (FFD) technique [39] was successfully extended to deform 3D face models by introducing a cylindrical structure in control lattices [12] or assigning weights to control points [23]. Singh and Fiume [41] present an effective geometric deformation technique to deform 3D face models by manipulating wire curves that are used as an approximation to the 3D face models. These approaches generally require significant manual efforts for tuning control points or curves in order to sculpt various facial expressions with fine details.

The blendshape approach (or shape interpolation) offers intuitive tools for sculpting 3D facial expressions [37, 10, 30, 38, 29, 40, 14]. For example, some recent efforts attempt to improve the efficiency of producing muscle actuation based blend shape animations [10, 40]. The Pose Space Deformation (PSD) approach proposed by Lewis *et al.* [30] provides a general framework for example-based interpolation where the deformation of a 3D surface (face) is treated as a function of a set of abstract parameters, such as {*smile, raise-eyebrow,...*}, and a new surface is generated by scattered data interpolations. However, these approaches generally require considerable manual efforts to create a set of blendshape targets.

Essentially, the above geometric deformation approaches and the blendshape methods are designed to simultaneously move and edit a group of relevant vertices. However, different facial regions are correlated each other, and the above approaches typically operate a local facial region or global shape at one time. Animators need to switch editing operations on different facial regions in order to sculpt realistic 3D faces with fine details, which create a large amount of additional work. In addition, even for skilled animators, it is difficult to judge which facial pose (configuration) is closer to a real human face. A number of data-driven facial editing techniques [6, 21, 29, 11, 46, 44, 31] were proposed to address this issue. For example, based on a blendshape representation for 3D face models, Joshi *et al.* [21] present an interactive tool to edit 3D face geometry by learning the optimal way for a physically-motivated face segmentation. A rendering algorithm for preserving visual realism in this editing was also presented in their approach. Editing a local facial region while preserving naturalness of the whole face is another intriguing idea. The geometry-driven editing technique [46] generates expression details on 2D face images by constructing a PCA-based hierarchical face representation from a selected number of training 2D face images. When users move one or several points on a 2D face image, the movements of other facial control points are automatically computed by a motion propagation algorithm.

Generating character animations and 3D facial expressions through sketching has been also explored by a number of researchers [13, 19, 7, 24, 26]. For example, Chang and Jenkins [7] present a 2D sketch interface for posing 3D faces. In their work, users can intuitively draw 2D strokes that are used to search for the optimal face pose. The work of [13, 24] presents sketching interfaces for efficiently prototyping articulated character animations.

3 Facial Motion Acquisition and Processing

We captured high-fidelity 3D facial motions using a VICON motion capture system (the left panel of Fig. 3). An actress with markers on her face was directed to speak a delicately designed corpus

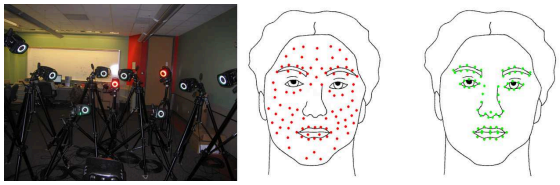

Figure 3: The left shows a facial motion capture system. The middle panel shows the facial markers used in this work. The right panel shows the 71 portrait control points on a human face.

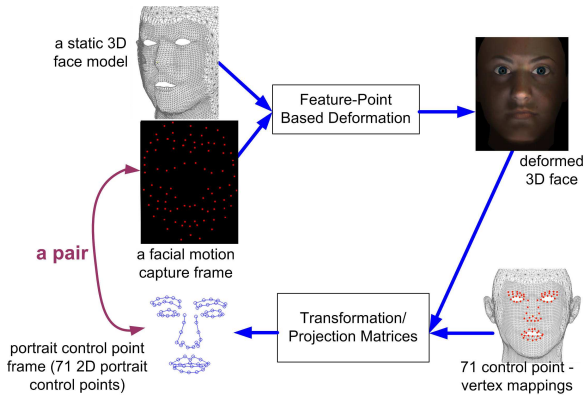

Figure 4: Illustration of how to generate corresponding portrait control point frame (seventy-one 2D portrait control points) from a 3D motion capture frames composed of ninety 3D facial markers.

four times, and each repetition was spoken with a different facial expression. In this data capture, a total of four basic facial expressions were recorded: *neutral, happiness, anger and sadness*, and the data recording rate was 120 Hz. A total of ninety facial markers (the middle panel of Fig. 3) were used in this work.

After data capture, we normalized the facial motion data by removing head motion as follows. All the markers were first translated so that a specific marker was at the local coordinate center of each frame, and then a Singular Value Decomposition (SVD) based approach [5] was used to remove head motion.

However, in the follow-up 2D portrait rendering algorithm (Section 4), seventy-one 2D portrait control points (illustrated in the right panel of Fig. 3) are needed as one of its inputs. Most of these seventy-one portrait control points are on the tight boundaries of eyebrows, eyes, and the mouth. It should be noted that the seventy-one portrait control points (the right panel of Fig. 3) are not exactly enclosed in the ninety facial markers (the red points in middle panel of Fig. 3). Hence, we need to generate corresponding seventy-one 2D portrait control points for any 3D facial motion capture frame (the ninety 3D facial markers). As shown in Fig. 4, first, based on a motion capture frame and the specified correspondences between markers and vertices, the feature point based deformation technique [25] is used to deform a static 3D face model. Then, given specified mappings between seventy-one portrait control points and the vertices of 3D face geometry, the 3D positions of these specified seventy-one vertices on the deformed 3D face are transformed and projected to a 2D plane, which outputs corresponding seventy-one 2D portrait control points. In this way, a pair between a 3D facial motion capture frame and its corresponding portrait control point frame was created. These pairs will be used in Section 7. These seventy-one 2D control points were referred collectively as *a portrait control point frame*.

4 2D Portrait Rendering

In this section we briefly describe the algorithm to render a 3D face model to a line drawing portrait. Portraiture is one of the most important genres in art [4]. However, there appears to be little research that addresses issues specific to automatic portrait rendering. Chen *et al.* generate frontal portraits from face photographs mimicking the style of a particular artist [9] or line drawings of frontal faces in a "manga" style using a data-driven scheme [8]. The creation of an interactive interface for non-photorealistic rendering [22] and the automated generation of pencil drawings from 3D models [42] were also proposed.

Although many image-based techniques (*e.g.*, artistic filters in Adobe Photoshop) are capable of transforming photographs into stylized drawings, they are unable to properly handle face photographs. This is due to the facts that: 1) image-based techniques place strokes without understanding what the subjects are; 2) as humans we are very sensitive to the faces of our own kind, and even a small mistake in a portrait (like a misplaced stroke) may destroy the likeness and even faceness.

It is known that artists are heavily guided by their prior knowledge of face structure. This general strategy is adopted in the portrait rendering process of this work: instead of treating a face as a generic image, we use a prior model of the face to locate and emphasize important face features and deemphasize unimportant details. A model-based method [32] is used to render 2D portraits in this work. This approach enables the generation of attractive line renderings of faces while offering a range of stylization and simplification options. The portrait rendering process is schematically illustrated in Fig. 5.

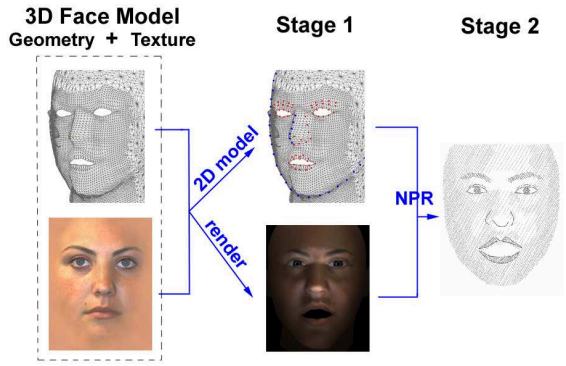

Figure 5: Schematic illustration of the portrait rendering process used in this work.

In stage 1, first, as shown in Fig 4, seventy-one feature points outlining eyebrows, eyes, nose, and lips are computed from a given 3D face model (geometry and texture). The 3D coordinates of these seventy-one points are view-invariant, and this is performed one time per 3D face model. Second, given a specific view, seventy-one feature points are projected into an image plane (red dots on the face model in Fig 5), and face and nose silhouette lines (view-dependent) are calculated using the same algorithm as described in [22] (blue dots on the face model in Fig. 5). Thus we obtain a complete 2D face model (*i.e.*, feature points as shown in Fig 5).

In stage 2, the process of portrait rendering is conceived as a transformation from a general image-derived description to a simplified and stylized representation of a face, guided by consulting the obtained face model. The input is the obtained 2D face model and the face image rendered from the 3D model, the output is a line drawing portrait.

A generic line stroke set is obtained by detecting edges (at several scales) in the rendered face photograph. Each stroke is assigned

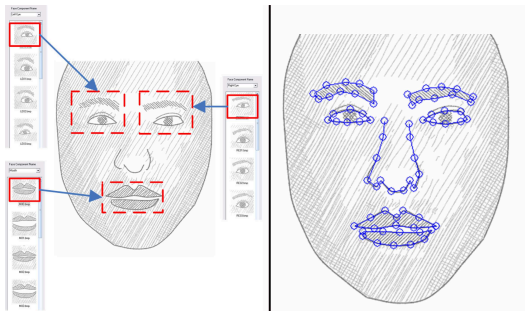

Figure 6: The left panel illustrates the Portrait Component Library (PCL) and the right panel shows the portrait control points. In the left panel, users select portrait components from the portrait component library. In this figure, three portrait components (for left eye, right eye and the mouth, respectively) are assembled to a complete 2D portrait.

an importance by consulting the model. The strokes are then filtered according to their importance and their length and strength. The level of detail or simplification can be controlled by adjusting thresholds on the stroke filtering.

5 Portrait Component Library

The purpose of Portrait Component Library (PCL) is to provide a rapid prototyping tool to assemble a portrait by *simply clicking*. For a prominent part of a human face, such as left eye area (including eyebrow), users can simply browse and pick one from a set of pre-defined left eye portrait components. These selected portrait components (one from each category) are assembled into a new 2D portrait that is typically used as an initial portrait for further editing. Figure 6 illustrates the portrait component library.

Now we describe how to construct the above portrait component library. As mentioned in Section 3, given the collected motion capture data, a large number of 2D portrait control point frames are generated. For a portrait control point frame, the 2D portrait control points that lie in a component area (*e.g.*, the mouth area) are concatenated to form a vector. Then the K-means clustering algorithm [20] is applied to these vectors to find their cluster centers, which are regarded as representatives of this category of portrait component. Finally, these representative portrait points are rendered using the above portrait rendering algorithm (Section 4). In this work, we experimentally divide a portrait into three components (left eye area, right eye area, and the mouth area) and set K (the number of clusters) to 10.

6 2D Portrait Editing

Given an existing 2D portrait, either default or assembled from the portrait component library, users are able to edit it by moving one or multiple portrait control points (the right panel of Fig. 6). When portrait control points are edited, the motion propagation algorithm proposed in [46] is adapted to automatically adjust the edited portrait to maximally maintain its naturalness and faceness.

We use a hierarchical principle component analysis [46] for this motion propagation. Essentially, we first divide the face into region nodes, then, we compute PCA separately for each node. Next, we form the tree from these nodes in order to propagate the motion projection from a leaf node to the whole tree at run-time. Initially, we divide the portrait into seven leaf nodes (left eye, left eyebrow, right eye, right eyebrow, nose, upper lip, and lower lip), and then construct two intermediate nodes (the upper face and the lower face). Finally, the whole portrait is regarded as the root of this hierarchy. Fig. 7 shows this portrait hierarchical structure.

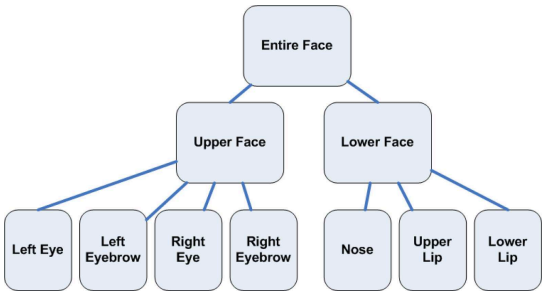

Figure 7: Illustration of the portrait hierarchy for motion propagation.

The rules for the above motion propagation procedure are 1) it chooses to move upward prior to downward in the hierarchy and 2) it visits each node only once. The propagation works as follows: first, when the users move portrait control points, the propagation starts at the lowest hierarchy which is one of the seven left-nodes. Then, it propagates upward to the middle of the hierarchy (upper or lower face nodes). Then, it moves upward to the root node that is the entire face. After that, it moves downward again to the middle node that it has not visited yet and it keeps going upward and downward until all nodes are visited. For each node it visits, it projects the control points contained in the node to the subspace spanned by the principal components of the node. In other words, the projection is the best approximation of the propagated motions in the PCA subspace of the node.

We precomputed a eigen-vector matrix $EigMx$ and a mean vector $MEAN$ from 2D portrait control points of each hierarchical node (Fig. 7). As such, each node in the hierarchy holds its own version of $EigMx$ and $MEAN$. In our experiment, to cover at least 90% of the variation, we keep the largest 20 eigenvectors for the entire face node, 10 for the upper and the lower face nodes, and 3 for each of the seven leaf nodes.

To make this section readable, we briefly summarize the basic steps of the motion propagation algorithm [46] in Algorithm 1. In the following algorithm description, F represents any node in the hierarchy, δV represents the displacement vector of all control points, and $Proj(\delta V, F*)$ denotes the projection of the $F*$ part of δV to the truncated PCA space of the node $F*$.

Algorithm 1 PortraitMotionPropagation

Input: $F*$, the selected node in the hierarchy.

1: set h to the hierarchy level of $F*$.;
2: **if** *hasBeenProcessed*($F*$) **then**
3: return;
4: **end if**
5: Compute *Proj*(δV,$F*$);
6: Update δV with *Proj*(δV,$F*$);
7: Set *hasBeenProcessed*($F*$) to be true;
8: **for** $\forall\ F \subseteq (level(F) = h-1$ and $F \cap F* = NonEmpty)$ **do**
9: *PortraitMotionPropagation*(F);
10: **end for**
11: **for** $\forall\ F \subseteq (level(F) = h+1$ and $F \cap F* = NonEmpty)$ **do**
12: *PortraitMotionPropagation*(F);
13: **end for**

Figure 8 shows an example of the portrait editing when users move a control point on the left eyebrow. As we can see from this figure, other control points are adjusted accordingly. It is noteworthy that the adjustments of the other control points in the left eyebrow area are noticeable.

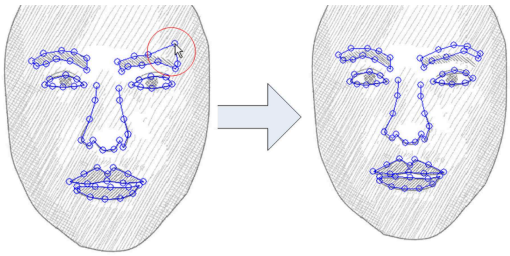

Figure 8: An example of motion propagation for portrait control points. Note that the back portrait (not updated) is used as a reference.

7 3D Facial Expression Generation

In this section, we describe how to generate corresponding 3D facial expression based on an edited 2D portrait. In this work, we transform this 3D facial expression generation problem to an optimal search problem. Given the fact that a 2D portrait is essentially determined by the seventy-one 2D portrait control points (of a portrait control point frame), and all pairs of portrait control point frames and 3D motion capture frames are precomputed in Section 3, we are able to formalize this search problem: given a portrait control point frame P_{query} (query input), we search for the optimal 3D facial motion capture frame FRM^* from all pairs $\{< P_i, FRM_i >\}$ in the database, assuming distance(P_{query},P^*) is minimum among all possible distances $\{distance(P_i, P_{query})\}$.

Considering the large size of the collected facial motion capture data, we choose a KD-tree [33] as our data structure for the above search due to its efficiency. Our KD-tree scheme consists of an off-line preprocessing and an on-line searching. In the off-line preprocessing stage, we reduce the dimensionality of the concatenated seventy-one 2D points (total 142 dimensions) to 10 dimensions in its truncated PCA space, while retaining more than 95% of variation. Meanwhile, we need to convert P_{query} to ten dimensional PCA_{query} by projecting P_{query} to the same PCA space. Figure 9 illustrates the constructed KD-tree in this system. In this work, the Simple KD-tree library [34] was used.

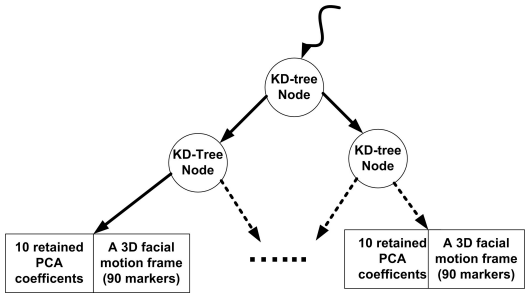

Figure 9: Illustration of the KD-tree structure in this work.

At run-time, the PCA_{query} is used as a query input to search through the constructed KD-tree to obtain the K nearest neighbors. We interpolate the searched K 3D motion capture frames to generate a new 3D motion capture frame that is used to deform a static 3D face model. The interpolation weights are computed based on the Euclidean distance between the retained PCA coefficients of the query input and the retrieved K nearest neighbors. In this work, we experimentally set K=3. Finally, given a 3D facial motion capture frame, the feature-point based deformation technique [25] is used to deform a static 3D face model accordingly.

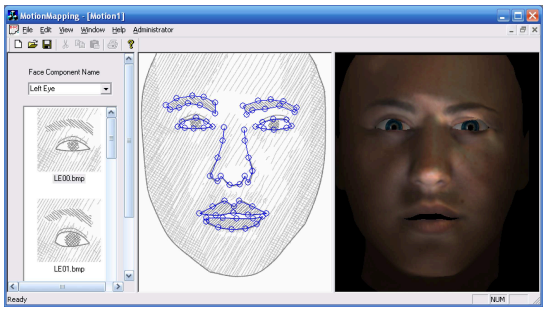

Figure 10: A snapshot of this running facial expression posing system.

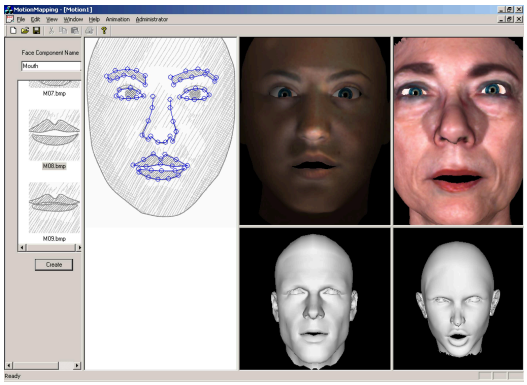

Figure 11: Facial expressions on various 3D face models are simultaneously sculpted based on the portrait (middle panel) that is being interactively manipulated.

8 Results and User Studies

We developed this 3D facial expression posing system using VC++. Figure 10 shows a snapshot of the running system. As shown in Fig. 10, its interface is composed of three panels: the portrait component library (left), a 2D portrait window (middle), and a generated 3D face window (right). Users can drag desired portrait components from the library (left) to the portrait window, and then edit the assembled or initial portrait. Meanwhile, the 3D face view window (right) will be interactively updated based on the 2D portrait (middle) that is being interactively edited. In addition, multiple 3D face models can be edited simultaneously in this system (Figure 11).

After 3D facial expressions are generated, users can view and manipulate them from different angles. One advantage of this new approach is that certain face details are automatically sculpted on 3D facial expressions, although they cannot be explicitly specified in 2D portraits. Figure 12 shows a specific example.

We conducted a comparative usability study on this system. A total of ten human subjects were asked to use both the Maya software and this system for the task of sculpting target 3D facial expressions. Since our target group is non-artist users but the people who is able to manipulate the computer in a good level, all the participants are computer science undergraduate or graduate students. They have an intermediate level skill of using basic tools of Maya, and were also trained to use this new system for one minute before the official start of this user study.

The scenario of this study was designed as follows:

1. Each participant is assigned four target facial expressions (2D images).

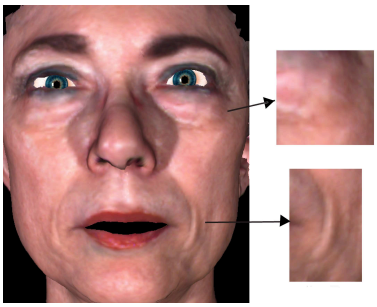

Figure 12: 3D face details are automatically generated using this system.

Figure 13: The first example of our user study. The top left is the target facial expression image. The top right is the sculpted 3D face expression using the Maya by a participant. The bottom is the posed 3D facial expression using this proposed system by the same participant.

2. Each participant need to sculpt 3D facial expressions that are sufficiently close to the given target facial expression images, using this system and the Maya software respectively.

3. A professional animator judges whether the sculpted 3D facial models by the two tools (this system and the Maya software) are acceptable (*i.e.*, close enough).

4. We record the time that each participant spend to get the sculpting task done using the two specified tools.

The average time using this system is less than 2 minutes for each 2D face image, and the average time using the Maya is about 27 minutes for each 2D face image. Our experimental result revealed that compared with traditional 3D tools such as the Maya, this 3D facial expression posing system can significantly save users' efforts for the task of sculpting 3D facial expressions. Figures 13 and 14 show two examples of sculpted facial expressions by the participants in this study.

In addition, at the end of the user study, we collected feedback from the participants regarding the usability of this system. A part of the feedback is as follows: (1) by using this system, they spend less time to deal with the 3D navigation gadget (*e.g.*, zooming and rotation) to manipulate 3D vertices and meshes, which helps them focus on the shapes of the 3D faces they are working on. (2) In the course of interactive sculpting, the intermediate facial expression

Figure 14: The second example of our user study. The top left is the target facial expression image. The top right is the sculpted 3D face model using the Maya by a participant. The bottom is the posed 3D facial expression using this proposed system by the same participant.

results produced by this system appear more natural than the Maya. This reduces significant repetitive efforts of re-adjusting 3D face models for them. (3) The Portrait Component Library (PCL) of this system is useful for them to get a good start and save their time.

We also generated numerous 3D facial expression results using this system. Figures 1and 15 show the resulting 2D portraits (top row) and corresponding 3D facial expressions on different 3D face models (other rows).

9 Discussion and Conclusions

In this paper, we present an interactive, data-driven 3D facial expression posing system through 2D portrait manipulation. This system is built on top of a pre-recorded facial motion capture database. This system allows users intuitively edit 2D portraits and then automatically generates corresponding 3D facial expressions with fine details. The 2D portrait allows users to focus on the face features and provides a more intuitive manipulation mechanism over traditional 3D editing systems. By conducting a comparative user study, we showed that this system can be effectively used as a rapid prototyping tool for 3D facial expression generation.

Certain limitations exist in this current system. First, since we use a pre-recorded facial motion dataset, it is hard to predict how much data we need to collect to guarantee the generation of realistic 3D facial expressions for arbitrary 2D portrait input. However, this can be alleviated by acquiring more data if needed. Second, the number of the available portrait components and the number of portrait control points are still quite limited. Adding more portrait details into the system is one of the future directions that we plan to pursue.

In the future, we plan to improve its 2D portrait editing procedure. For example, it will allow users edit portraits from different viewing angles, rather than limited to the front view in current system. Another area that we will look into is to develop a free-hand 2D portrait drawing tool and make the system more intuitive and friendly to users, especially novice users. In addition, current system can only interpolate new facial expressions from the existing facial motion dataset and cannot deal with exaggerated portraits. As future work, we plan to develop a portrait exaggeration function and seamlessly convert the exaggerated portraits to corresponding 3D facial expressions by a hybrid of machine learning and geometric deformation algorithms.

Figure 15: Six edited portraits (top row) and corresponding generated 3D facial expressions (other rows, each row for a different 3D face model).

10 ACKNOWLEDGEMENTS

Tanasai Sucontphunt was supported in part by fellowship from the Ministry of Science and Technology, Royal Thai Government. We also would like to thank George Toderici and Jose Baez-Franceschi for building 3D face models used in this work.

REFERENCES

[1] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *Proc. of SIGGRAPH'99*, pages 187–194, 1999.

[2] M. Brand. Voice pupperty. In *Proc. of SIGGRAPH'99*, pages 21–28, 1999.

[3] C. Bregler, M. Covell, and M. Slaney. Video rewrite: Driving visual speech with audio. In *Proc. of SIGGRAPH'97*, pages 353–360, 1997.

[4] R. Brilliant. *Portraiture*. Reaktion Books, 2001.

[5] C. Busso, Z. Deng, U. Neumann, and S. Narayanan. Natural head motion synthesis driven by acoustic prosody features. *Computer Animation and Virtual Worlds*, 16(3-4):283–290, July 2005.

[6] Y. Cao, P. Faloutsos, and F. Pighin. Unsupervised learning for speech motion editing. In *SCA'03: Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003.

[7] E. Chang and O. Jenkins. Sketching articulation and pose for facial animation. In *SCA'06: Proc. of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2006.

[8] H. Chen, Z. Liu, C. Rose, Y. Xu, H.-Y. Shum, and D. Salesin. Example-based composite sketching of human portraits. In *NPAR'04: Proc. of the 3rd International Symopsium on Non-Photorealistic Animation and Rendering*, pages 95–153, 2004.

[9] H. Chen, Y.-Q. Xu, H. Shum, S. Zhu, and N. Zheng. Example-based facial sketch generation with nonparametric sampling. In *ICCV 2001*, pages 433–438, 2001.

[10] B. W. Choe and H. S. Ko. Analysis and synthesis of facial expressions with hand-generated muscle actuation basis. In *Proc. of IEEE Computer Animation*, pages 12–19, 2001.

[11] E. S. Chuang, H. Deshpande, and C. Bregler. Facial expression space learning. In *Proc. of Pacific Graphics'02*, pages 68–76, 2002.

[12] S. Coquillart. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. *Computer Graphics*, 24:187–193, 1990.

[13] J. Davis, M. Agrawala, E. Chuang, Z. Popović, and D. Salesin. A sketching interface for articulated figure animation. In *SCA '03: Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 320–328, 2003.

[14] Z. Deng, P. Chiang, P. Fox, and U. Neumann. Animating blendshape faces by cross mapping motion capture data. In *I3DG'06: Proc. of the 2006 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 43–48, 2006.

[15] Z. Deng and U. Neumann. eFASE: Expressive facial animation synthesis and editing with phoneme-isomap controls. In *SCA'06: Proc. of ACM SIGGRAPH/EG Symposium on Computer Animation*, pages 251–259, Vienna, Austria, 2006.

[16] Z. Deng and U. Neumann. *Data-Driven 3D Facial Animation*. Springer-Verlag Press, 2007.

[17] Z. Deng, U. Neumann, J. Lewis, T. Y. Kim, M. Bulut, and S. Narayanan. Expressive facial animation synthesis by learning speech co-articulation and expression spaces. *IEEE Transaction on Visualization and Computer Graphics*, 12(16):1523–1534, 2006.

[18] T. Ezzat, G. Geiger, and T. Poggio. Trainable videorealistic speech animation. *ACM Trans. Graph.*, pages 388–398, 2002.

[19] B. Gooch, E. Reinhard, and A. Gooch. Human facial illustrations: Creation and psychophysical evaluation. *ACM Trans. Graph.*, 23(1):27–44, 2004.

[20] T. Hastie, R. Ribshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[21] P. Joshi, W. Tien, M. Desbrun, and F. Pighin. Learning controls for blend shape based realistic facial animation. In *SCA'03: Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 35–42, 2003.

[22] R. Kalnins, L. Markosian, B. Meier, M. Kowalski, J. Lee, P. Davidson, M. Webb, J. Hughes, and A. Finkelstein. WYSIWYG NPR: drawing strokes directly on 3D models. In *Proc. SIGGRAPH 2002*, pages 755–762, 2002.

[23] P. Kalra, A. Mangili, N. Thalmann, and D. Thalmann. Simulation of facial muscle actions based on rational free from deformations. In *Eurographics'92*, volume 11, pages 59–69, 1992.

[24] O. A. Karpenko and J. F. Hughes. Smoothsketch: 3D free-form shapes from complex sketches. *ACM Trans. Graph.*, 25(3):589–598, 2006.

[25] S. Kshirsagar, S. Garchery, and N. M. Thalmann. Feature point based mesh deformation applied to mpeg-4 facial animation. In *Proc. Deform'2000, Workshop on Virtual Humans by IFIP Working Group 5.10*, pages 23–34, November 2000.

[26] M. Lau, J. Chai, Y.-Q. Xu, and H.-Y. Shum. Face poser: interactive modeling of 3D facial expressions using model priors. In *SCA'07: Proc. of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 161–170, 2007.

[27] K. R. Laughery and R. H. Fowler. Sketch artist and identi-kit procedure for recalling faces. *Journal of Applied Psychology*, 65(3):307–316, 1980.

[28] Y. Lee, D. Terzopoulos, and K. Waters. Realistic face modeling for animation. In *Proc. of SIGGRAPH'95*, pages 55–62, 1995.

[29] J. Lewis, J. Mooser, Z. Deng, and U. Neumann. Reducing blendshape interference by selected motion attenuation. In *I3DG'05: Proc. of the 2005 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 25–29, 2005.

[30] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proc. of SIGGRAPH'00*, pages 165–172, 2000.

[31] Q. Li and Z. Deng. Facial motion capture editing by automated orthogonal blendshape construction and weight propagation. *IEEE Computer Graphics and Applications*, 2008, (to appear).

[32] Z. Mo, J. Lewis, and U. Neumann. Improved automatic caricature by feature normalization and exaggeration. In *SIGGRAPH 2004 Sketches*, 2004.

[33] A. Moore. An intoductory tutorial on KD-trees. *PhD Thesis: Efficient Memory based Learning for Robot Control, PhD Thesis Technical Report No. 209*, 1990. University of Cambridge.

[34] A. Moore and J. Ostlund. Simple kd-tree library. *http://www.autonlab.org/*, 2007.

[35] J. Y. Noh and U. Neumann. Expression cloning. In *Proc. of SIGGRAPH'01*, pages 277–288, 2001.

[36] F. Parke and K. Waters. *Computer Facial Animation*. 1996.

[37] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. Salesin. Synthesizing realistic facial expressions from photographs. In *Proc. of SIGGRAPH'98*, pages 75–84, 1998.

[38] H. Pyun, Y. Kim, W. Chae, H. W. Kang, and S. Y. Shin. An example-based approach for facial expression cloning. In *SCA'03: Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 167–176, 2003.

[39] T. Sederberg and S. Parry. Free-form deformation of solid geometry models. In *Proc. of SIGGRAPH'96*, volume 20, pages 151–160, 1996.

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

[41] K. Singh and E. Fiume. Wires: A geometric deformation technique. In *Proc. of SIGGRAPH 98*, pages 405–414, 1998.

[42] M. Sousa and J. Buchannan. Computer-generated pencil drawing. In *Proc. SKIGraph'99*, 1999.

[43] M. Viad and H. Yahia. Facial animation with wrinkles. In *Proc. of Eurgraphics Workshop on Animation and Simulation*, 1992.

[44] D. Vlasic, M. Brand, H. Pfister, and J. Popović. Face transfer with multilinear models. *ACM Trans. Graph.*, 24(3):426–433, 2005.

[45] C. Wang and D. R. Forsey. Langwidere: A new facial animation system. In *Proc. of IEEE Computer Animation*, pages 59–68, 1994.

[46] Q. Zhang, Z. Liu, B. Guo, and H. Shum. Geometry-driven photorealistic facial expression synthesis. In *SCA'03: Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 177–186, 2003.