DOI: 10.1111/j.1467-8659.2008.01192.x

# Expressive Speech Animation Synthesis with Phoneme-Level Controls

Z. Deng[1] and U. Neumann[2]

[1]Computer Graphics and Interactive Media Lab, Department of Computer Science University of Houston, Houston, TX, USA
zdeng@cs.uh.edu
[2]Department of Computer Science, University of Southern California, Los Angeles, CA, USA
uneumann@graphics.usc.edu

**Abstract**

*This paper presents a novel data-driven expressive speech animation synthesis system with phoneme-level controls. This system is based on a pre-recorded facial motion capture database, where an actress was directed to recite a pre-designed corpus with four facial expressions (neutral, happiness, anger and sadness). Given new phoneme-aligned expressive speech and its emotion modifiers as inputs, a constrained dynamic programming algorithm is used to search for best-matched captured motion clips from the processed facial motion database by minimizing a cost function. Users optionally specify 'hard constraints' (motion-node constraints for expressing phoneme utterances) and 'soft constraints' (emotion modifiers) to guide this search process. We also introduce a phoneme–Isomap interface for visualizing and interacting phoneme clusters that are typically composed of thousands of facial motion capture frames. On top of this novel visualization interface, users can conveniently remove contaminated motion subsequences from a large facial motion dataset. Facial animation synthesis experiments and objective comparisons between synthesized facial motion and captured motion showed that this system is effective for producing realistic expressive speech animations.*

**Keywords:** Facial animation, speech animation, data-driven, facial expression, phoneme-isomap, motion capture

**ACM CCS:** I.6.8 [Computing Methodologies]: Simulation and modelling – types of simulation; I.3.7 [Computing Methodologies]: Computer graphics – three dimensional graphics and realism

## 1. Introduction

Synthesizing realistic and human-like facial animations is still one of the most challenging topics in the computer graphics and animation community. Manual approaches typically pose keyframes every several frames to create compelling facial animations, which is a painstaking and tedious task, even for skilled animators. Facial motion capture, widely used in entertainment industry, can acquire high-fidelity human facial motion data; however, it remains difficult to modify and edit captured facial motion data to generate novel facial animations.

In this work, we present a novel data-driven expressive speech animation synthesis system by searching for best-matched motion capture frames in a pre-recorded facial motion database while animators establish constraints and goals. The inputs of this system include novel speech with phoneme-alignment information, and optional user-specified constrained expressions for phonemes and emotions that are used only to impart them with a desired expressiveness. On the user interaction and control side, this system offers flexible and intuitive phoneme-level controls to users. Users can browse and select constrained expressions for phonemes using a two-dimensional (2D) expressive phoneme–Isomap visualization interface introduced in this work. Meanwhile, users can optionally specify emotion modifiers over arbitrary phoneme time. These speech-related controls are performed on a phoneme level. Figure 1 illustrates high-level components of this system.

This system provides an intuitive tool for browsing and eliminating contaminated motion subsequences from a large facial motion dataset. There are two reasons why this tool

Submitted December 2007
Revised April 2008
Accepted April 2008

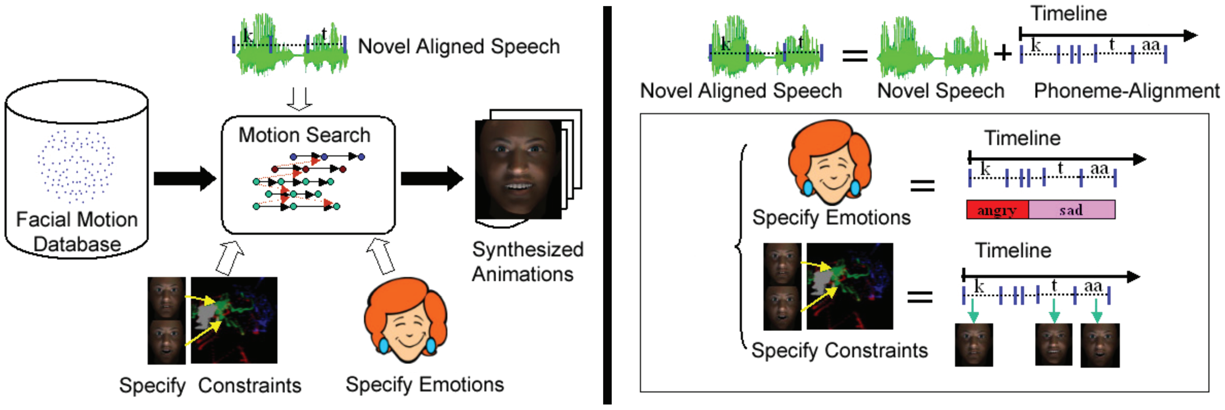


**Figure 1:** *Schematic illustration of the expressive speech animation generation system. At the left, given novel phoneme-aligned speech and (optional) specified constraints, this system searches for best-matched motion nodes in the facial motion database and synthesizes expressive speech animation. The right panel illustrates how users specify motion-node constraints and emotions with respect to the speech timeline.*

is useful. First, intuitively visualizing and managing a large facial motion dataset that is composed of hundreds of thousands motion frames is a necessary step towards its various applications. Second, acquired facial motion capture data often is imperfect, and contaminated marker motions can occur somewhere in a motion capture sequence. Eliminating these contaminated motion subsequences is difficult but very useful. The phoneme Isomap based browsing and checking tool offers an intuitive way to help users to remove contaminated motion sequences efficiently. A preliminary version of this work has been published as a conference paper [DN06].

The contributions of this work include:

(i) *Its expressive speech motion synthesis algorithm* is an improvement over previous data-driven synthesis algorithms. Emotion controls (modifiers) introduced in this work can seamlessly synthesize expressive speech animations with four facial expressions (neutral, happiness, anger and sadness) captured in our dataset.
(ii) *Intuitive phoneme-level controls* allow users to interact with its speech animation synthesis procedure conveniently via the 2D visualization of phoneme Isomaps. The phoneme–isomap based interface provides an intuitive tool for browsing and eliminating contaminated motion subsequences from a large facial motion dataset, which is difficult to achieve using traditional approaches.

The remainder of this paper is organized as follows: Section 2 reviews previous and related work on motion capture and facial animation synthesis. Section 3 describes how expressive facial motion data are acquired and processed. Section 4 describes the construction of 2D expressive phoneme Isomaps that allow users to interactively specify phoneme expression constraints and manage the large facial motion database. Section 5 details how to perform motion database management operations and how to specify constraints for expressing phonemes. Section 6 describes how to search for best-matched motion nodes from the processed motion-node database to create complete expressive speech animations while satisfying user-specified constraints. Section 7 describes how to real-time deform three-dimensional (3D) face models based on marker motion data and how to smooth searched motion frames. Finally, Section 8 describes experiment results and evaluations.

## 2. Previous and Related Work

Various facial modelling and animation techniques have been developed in the past several decades [DN07a, PW96]. For example, physically-based methods [KHS01, LTW95, WF95] drive facial and mouth movements by simulating the facial muscles. Performance-driven facial animation techniques [Wil90] deform 3D face models based on facial motion of real performers, tracked using computer vision techniques. The work of [GGW*98, PHL*98, ZSCS04] uses blendshapes or traverse faces modelled from photographs or video streams to generate facial animations. Blanz *et al.* [BV99] creates a morphable face model from a set of scanned 3D face model dataset, then reconstructs 3D faces from input 2D images and video and produce novel facial animations [BBPV03]. A number of techniques were also presented to transfer facial animations of source face models to other face models automatically [DCFN06, NN01, PKC*03, SNF05, VBPP05]. Refer to the comprehensive survey by Deng and Noh [DN07b] for recent advances in computer facial modelling and animation.

Phoneme-based methods use hand-crafted speech co-articulation models [BP04, CM93, Lew91, Pel91] or model speech co-articulation from data [DLN05, DNL*06, KP05].

The Cohen–Massaro co-articulation model [CM93] represents each viseme shape using a target value and a dominance function, and determines final mouth shapes based on the weighted sum of dominance values. Bevacqua and Pelachaud [BP04] presented an expressive qualifier, modelled from recorded real motion data, to make expressive speech animations. Deng *et al.* [DLN05, DNL*06] learn parametric speech co-articulation models and expression spaces from motion capture data for the facial animation synthesis purpose.

Data-driven facial animation approaches learn statistical models from data [Bra99, CB05, CDB02, CFP03, DNL*06, EGP02] for facial animation synthesis and editing. Brand [Bra99] learns HMM-based facial control models by an entropy minimization learning algorithm from voice and video training data and then effectively synthesizes full facial motions from novel audio track. Ezzat *et al.* [EGP02] learn a multidimensional morphable model from a recorded video database that requires a limited set of mouth image prototypes. Chuang *et al.* [CB05, CDB02] learn a facial expression mapping/transformation from training footage using bilinear models, and then this learned mapping is used to transform novel video of neutral talking to expressive talking. Cao *et al.* [CFP03] present a motion editing technique that applies independent component analysis (ICA) to recorded facial motion capture data and further editing and mapping operations are done on these ICA components, interpreted as expression and speech components separately. Zhang *et al.* [ZLGS03] present a geometry-driven technique for synthesizing expression details for 2D faces, where users can move a small number of control points on 2D face images, and then movements of other 2D control points are automatically computed using a motion propagation algorithm.

Another way of exploiting data for facial animation synthesis is to concatenate phoneme or syllable segments from a pre-recorded facial motion database [BCS97, CFKP04, CG00, EGP02, KT03, MCP*05]. Bregler *et al.* [BCS97] present the 'video rewrite' method for synthesizing 2D talking faces given novel audio track, based on a collected database of 'triphone video segments'. The success of this approach largely depends on the accuracy of phoneme-alignment and the number of triphones covered in the training video footage. Instead of constructing a phoneme segment database, Kshirsagar and Thalmann [KT03] present a syllable based approach to synthesize novel speech animation. In their approach, captured facial motions are chopped and categorized into syllable motions, and then new speech animations are generated by concatenating corresponding syllable motions from the created syllable motion database. Rather than restricting within triphones or syllables, longer ($\geq 3$) phoneme sequences can be combined in an optimal way using various search methods including greedy search [CFKP04] or the Viterbi search algorithm [CG00, MCP*05]. These above approaches can achieve synthesis realism, but their versatility and control are limited. One of their common limitations is that it is difficult to achieve facial expression (emotion) control over synthesis procedure without considerable efforts.

The expressive speech animation generation system presented in this work employs a constrained dynamic programming algorithm to search for the best-matched motion capture subsequences in a created motion-node database, similar to [CG00, MCP*05]. But the main distinctions of our search algorithm include: (i) By introducing a novel emotion control (modifier), our algorithm can seamlessly synthesize expressive facial animation, instead needing to create separate facial motion database for each emotion category, as previous approaches have done. (ii) It introduces motion-node constraints for expressing phonemes into the animation synthesis process, which make the control of data-driven facial animation synthesis feasible and intuitive.

Researchers in the graphics field also pursued ways of editing and reusing the captured body motion data while maintaining its intrinsic realism. Recent approaches synthesize human body motion by motion warping [WP95], a parameterized interpolation approach for generating different styles of human motion [RCB98], learning statistical models for motion styles [BH00], constructing and traversing motion graphs [AF02, KGP02], sampling from learned statistical models [LWS02, PB02], searching in optimized low dimensional manifolds [GMHP04, SHP04], or constructing a continuous parameterized space of motions [KG04].

In a broad sense, human body motion synthesis from body motion capture data has some connections with this work, but these methods can not be directly applied to expressive speech animation synthesis due to control differences. Body motions are more involved with interactions between bones, while human facial motions are non-rigid deformations among facial muscles, bones and skin. The non-rigid deformations impose great difficulty on controlling facial motion synthesis. This is evident in that a simple skeleton model is acceptable for representing synthesized body motion and key body poses, while simplified facial representations, such as markers or line drawings, are not usable for specifying keyframe face configurations or demonstrating animations without considerable efforts. As such, it makes *intuitive controls for facial animation synthesis* an important, while difficult, topic in the field. In this work, we attempt to tackle this control issue by introducing a novel phoneme–Isomap interface that provides users with convenient controls.

## 3. Facial Motion Data Capture and Process

To acquire high-fidelity 3D facial motion data, we captured facial motions of an actress using a VICON motion capture system (the left panel of Figure 2). The actress with 102 markers on her face was directed to speak a delicately designed corpus four times, and each repetition was spoken with a

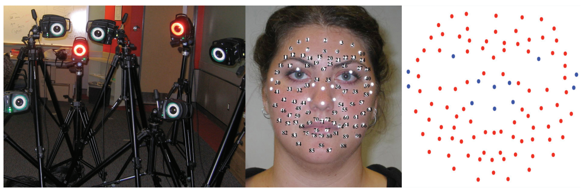


**Figure 2:** *The left shows a facial motion capture system, the middle is a snapshot of the captured actress. In the right panel, blue and red points represent the 102 captured markers, where the red points are the 90 markers used for this work.*

different facial expression. The corpus was designed as follows: we first collected the set of North American diphones (phoneme pairs) from the Carnegie Mellon University Pronouncing Dictionary [cmu05] and then designed sentences to cover the majority of the collected diphone set. In our work, the corpus consists of about 225 phoneme-balanced sentences. In this work, a total of four different expressions are considered: *neutral, happiness, anger and sadness*. It should be noted that sentences for each expression repetition are slightly different, because the actress had difficulty in reciting some sentences with certain expressions.

The facial motion data was recorded with a 120 frames/second rate. When facial motions were being captured, simultaneous acoustic speech was also recorded. We collected approximately 135 minutes of facial motion capture data. Because of tracking errors caused by rapid large head movements and the removal of unnecessary facial markers, we used only 90 of 102 markers for this work. The 90 markers were fully tracked. Figure 2 shows a facial motion capture system, the 102 captured markers and the 90 kept markers.

The motion frames for each corpus repetition were labelled with the intended emotion, the only tag information required by algorithms used in this work. Except for 36 sentences that were reserved for cross-validation and test comparisons, the other captured facial motion data were used for constructing the training facial motion database.

After data capture, we normalized the facial motion data. All the markers were translated so that a specific marker was at the local coordinate centre of each frame. Then, a statistical shape analysis method [BDNN05] was used to calculate head motion. A neutral pose with closed mouth was chosen as a reference frame and was packed into a $90 \times 3$ matrix, $y$. For each motion capture frame, a matrix $x_i$ was created using the same marker order as the reference. After that, the *singular value decomposition* (SVD), $UDV^T$, of matrix $y^T x_i$ was calculated. Finally, the product of $VU^T$ gave the rotation matrix, $R$.

$$y^T x_i = UDV^T \qquad (1)$$

$$R = VU^T. \qquad (2)$$

The *Festival* system [fes04] was used to perform automatic phoneme alignment on the captured audio in a forced-alignment mode. Accurate phoneme-alignment is important to the success of this work and automatic phoneme-alignment is imperfect, so two linguistic experts manually checked and together checked and corrected all the phoneme-alignments by examining the corresponding spectrograms.

After head motion was removed from the facial motion capture data, the motions of all 90 markers in one frame were packed into a 270 dimensional motion vector, and the principal component analysis (PCA) algorithm was applied to all the motion vectors to reduce their dimensionality. We experimentally set the reduced dimensionality to 25, which covers 98.53% of the motion variation. In this way, we transformed a 270-dimensional motion vector into a reduced 25-dimensional vector concatenating the retained PCA coefficients. In this work, we use *motion frames* to refer to these retained PCA coefficient vectors or their corresponding 3D facial marker configurations.

To make terms used in this paper consistent, we introduce two new terms: *motion nodes* and *phoneme clusters*. Based on the phoneme time boundaries, we chopped the motion capture sequences into subsequences that span several to tens of motion frames, and each subsequence corresponds to the duration of a specific phoneme. Each phoneme occurs many times in the spoken corpus, with varied co-articulations. We refer to these facial motion subsequences as *motion nodes*. It should be noted that in most of published facial animation literature, a viseme has been already used to represent the *static* visual representation of a phoneme. However, in this paper a motion node is a visual motion subsequence enclosing timing information. To differentiate it from the widely-used viseme, we use 'motion nodes'.

For each motion node, its triphone context that includes its previous phoneme and next phoneme is also retained. Putting all motion nodes of a specific phoneme together produces thousands of motion frames representing captured facial configurations that occur for the phoneme. All the motion frames corresponding to a specific phoneme are referred to as a *phoneme cluster*. Each motion-frame in a phoneme cluster has an emotion label and a normalized relative time property (relative to the duration of the motion node that it belongs to) that is in the range from 0 and 1. The specific phoneme that a motion node represents is called *the phoneme of the motion node*. Figure 3 illustrates the process of constructing phoneme clusters and motion nodes.

Besides the phoneme clusters, we also built a facial motion-node database. The processed motion node database can conceptually be regarded as a 3D space (spanned by *sentence*, *emotion* and *motion node order*). Because sentence is the atomic captured unit, each motion node $o_i$ (except the

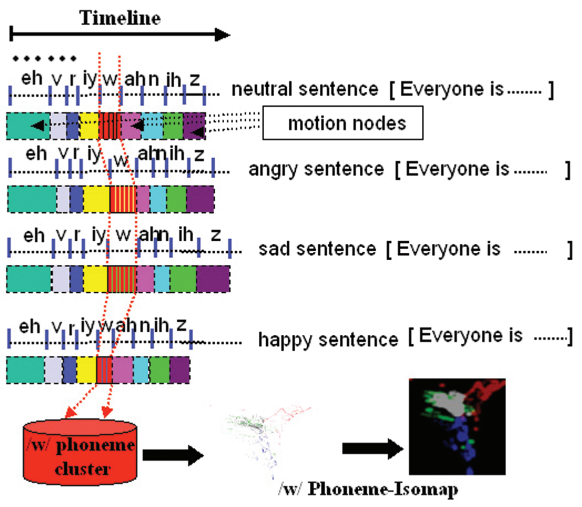

Timeline
eh v r iy w ah n ih z
neutral sentence [ Everyone is ........ ]
motion nodes
angry sentence [ Everyone is ........ ]
sad sentence [ Everyone is ........ ]
happy sentence [ Everyone is ........]
/w/ phoneme cluster
/w/ Phoneme-Isomap

**Figure 3:** *To construct a specific/w/phoneme cluster, all expressive motion capture frames corresponding to/w/ phonemes are collected, and the Isomap embedding generates a 2D expressive phoneme Isomap. Coloured blocks in the figure are motion nodes.*

first/last motion node of a sentence recording) has a predecessor motion node $pre(o_i)$ and a successive motion node $suc(o_i)$ in its sentence (illustrated as solid directional lines in Figure 4). Possible transitions from one motion node to another motion node are illustrated as dashed directional lines in Figure 4. A noteworthy point is motion nodes for the silence phoneme/pau/were discarded, and if the/pau/phoneme appears in the middle of a sentence's phoneme transcript, it will break the sentence into two sub-sentences when constructing this motion node database. Special post-processing for the silence phoneme/pau/will be described in Section 7.

Figure 4 illustrates the conceptual organization of the processed facial motion node database.

## 4. Expressive Phoneme Isomaps

Now we describe how to transform and visualize the phoneme clusters into 2D expressive phoneme Isomaps. The phoneme Isomaps enable users to interactively browse and select motion frames and motion nodes. The idea of creating intuitive visualization interfaces for phoneme clusters was largely inspired by Safonova *et al.*'s work [SHP04] where PCA is applied to a specific type of human body motion, e.g. *jumping*, to generate a low-dimensional manifold. In this work, each phoneme cluster is processed with the Isomap framework [TSL00] to embed the cluster into a 2D manifold. The neighbour number parameter is set to 12 in this work.

We compared 2D phoneme-PCA maps (two largest eigenvector expanded spaces) with 2D phoneme Isomaps. By visualizing both in colour schemes, we found that points for one specific colour (emotion) were distributed throughout the 2D PCA maps, while the 2D phoneme Isomaps cluster the same colour (emotion) points in a more natural way and lead to a better projection. As such, as a mean for frame selection, the 2D PCA display is not as good as the 2D phoneme Isomaps. We also found that certain directions in the phoneme Isomaps, such as a vertical axis, often corresponded to intuitive perceptual variations of facial configurations, such as an increasingly open mouth. Figure 5 compares 2D PCA projection and 2D Isomap projection on the same phoneme clusters.

The above point-rendering (Figure 5) of 2D expressive phoneme Isomaps are not directly suitable for interactively browsing and selecting facial motions. A Gaussian kernel point-rendering technique further visualizes the phoneme

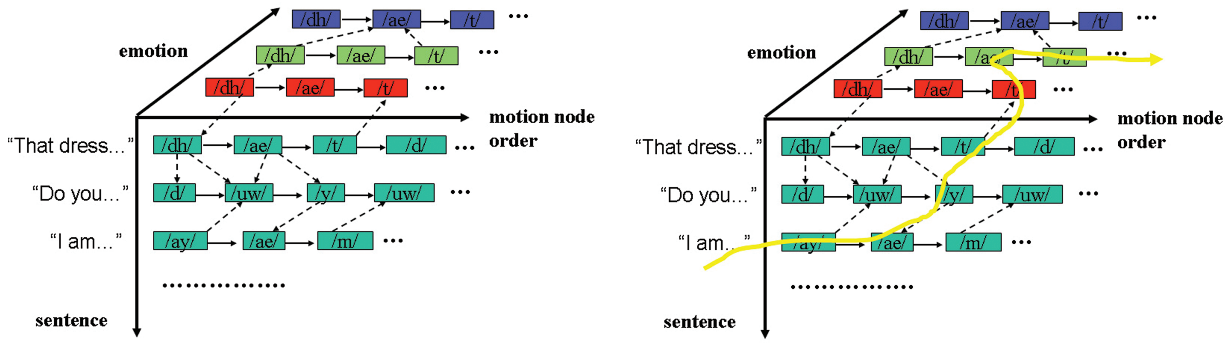


**Figure 4:** *Conceptual illustration of the constructed motion node database. Here, solid directional lines indicate predecessor/successor relations between motion nodes, and dashed directional lines indicate possible transitions from one motion node to the other. The colours of motion nodes represent different emotion categories of the motion nodes. The yellow line in the right panel represents a possible motion node path.*

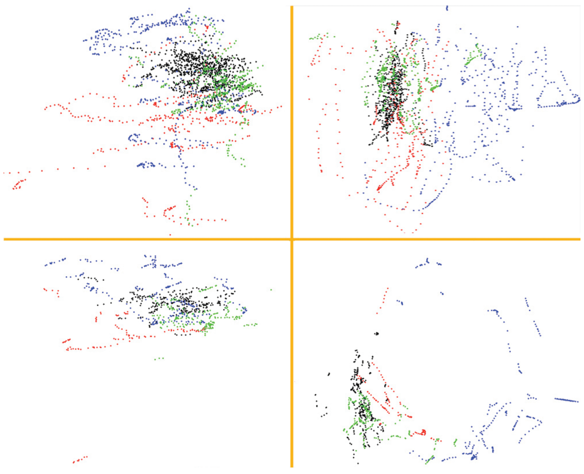


**Figure 5:** *Comparisons between 2D phoneme-PCA maps and 2D phoneme Isomaps. The left panels are 2D phoneme-PCA maps for / aa / (top) and / y / (bottom), and the right panels are 2D phoneme Isomaps for / aa / (top) and / y / (bottom). In all four panels, black is for neutral, red for angry, green for sad and blue for happy. Note that some points may overlap in these plots.*

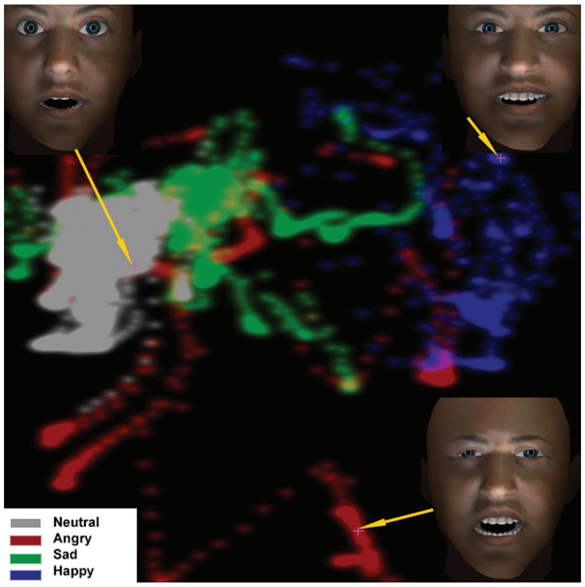


**Figure 6:** *Illustration of a 2D expressive phoneme Isomap for phoneme / ay /. Here, each point in the map corresponds to a specific 3D facial configuration.*

Isomaps as follows: each pixel of the phoneme Isomap accumulates the Gaussian distributions centred at each embedded location, and pixel brightness is proportional to the probability of a corresponding motion-frame representing the phoneme. Assume $P$ is a pixel on a 2D phoneme Isomap, the following Equation (3) is used to compute $I(P)$, the pixel brightness of $P$.

$$I(P) = \sum_{i=1}^{n} N(Q_i, \sigma; distance(Q_i, P)) * \rho * I(Q_i). \tag{3}$$

Here, $\rho$ is a user-defined positive value assigned to each facial frame in the phoneme cluster, and $0 < \rho < 1$, $Q_i$ is the embedded 2D location of one frame in the phoneme cluster, $N(.)$ is a user-defined Gaussian distribution centred at $Q_i$, $I(Q_i)$ (the pixel brightness of $Q_i$) is pre-assigned based on the chosen colouring scheme. If $I(P)$ is larger than one, then it is clamped to one. Figure 6 shows an example of the visualized phoneme Isomaps.

Then, we apply the 2D Delaunay triangulation algorithm to the embedded 2D Isomap coordinates of each phoneme Isomap to construct its triangulation network, where each vertex (of these triangles) corresponds to an embedded phoneme Isomap point – a facial motion frame in the phoneme cluster. These triangles cover most of the points in the phoneme Isomap image without overlap, except some points around the image boundary may not be covered by the triangulation network.

Therefore, when users pick a point $x$ in a phoneme Isomap, a unique covering triangle that covers $x$ is determined based on the precomputed triangulation network. Finally, the barycentric coordinates of $x$ are used to interpolate three vertices (motion-frames) of the covering triangle to generate a new motion-frame (corresponding to the picked point). Figure 7 illustrates this process.

As such, a phoneme Isomap image is a visualized representation of a continuous space of recorded facial configurations for one specific phoneme. Figure 6 shows an example of the phoneme Isomap images (for the/ay/phoneme). Note that these phoneme Isomap images and their mapping/triangulation information were precomputed and stored for later use. Based on the above-interpolated motion frame (Figure 7), a 3D face model is deformed correspondingly (refer to Section 7.1 for more details).

## 5. Browse and Manage Facial Motion Nodes

As described in Section 3, the captured facial motion database is generally composed of hundreds of thousands of facial motion capture frames, and it is challenging to manage this huge dataset. The phoneme Isomap images allow users to browse and manage these motion nodes intuitively and conveniently.

Each motion node is a sequence of motion capture frames of one specific phoneme in their recorded order, which is visualized as a directed trajectory in the phoneme Isomap images. As each point on the trajectory represents a specific

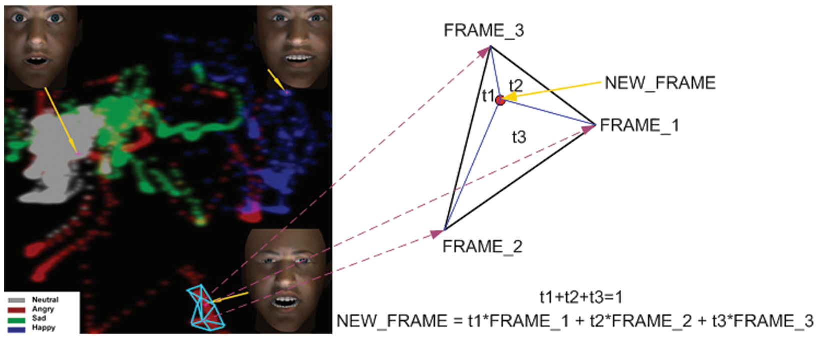


**Figure 7:** *Illustration of how a picked point is transformed to a new facial motion frame.*

facial configuration (refer to Figure 6), and the pixel colour behind a motion-node trajectory represents the emotion category of the motion node, users can intuitively and conveniently inspect any frame in the motion node (a point on the trajectory) as follows: when users click any point on the trajectory, its corresponding 3D face deformation is interactively displayed in a preview window. Besides offering single motion frame preview, this system can be straightforwardly extended to handle previewing ‘expressive facial motion clips’ as follows: if users select a motion node (trajectory) in a phoneme Isomap image, a clip preview window can show the animation of the corresponding motion node.

During this interaction process, if contaminated motion nodes are found, users can easily remove these motion nodes from the database. The motion synthesis algorithm (Section 6) could avoid the risk of being trapped into these contaminated motion nodes. Furthermore, other facial motion database based applications will benefit from a cleaned facial motion database.

## 6. Expressive Speech Motion Synthesis

In this section, we describe how the motion synthesis algorithm in this work automatically synthesizes corresponding expressive speech/facial animations, given a novel phoneme sequence (extracted from speech/texts input) and its emotion specifications. This system is fully automatic after inputs are fed into the system. However, in many animation applications, providing intuitive user controls is as important as full automation. This system provides intuitive user controls in two ways: users can specify a motion-node constraint for any phoneme utterance (hard constraints) via the above phoneme–Isomap interface, and specify the emotion modifiers as ‘soft constraints’. Under these hard and soft constraints, the motion synthesis algorithm in this work searches for a best-matched sequence (path) of motion nodes from the processed facial motion node database, by minimizing a cost function.

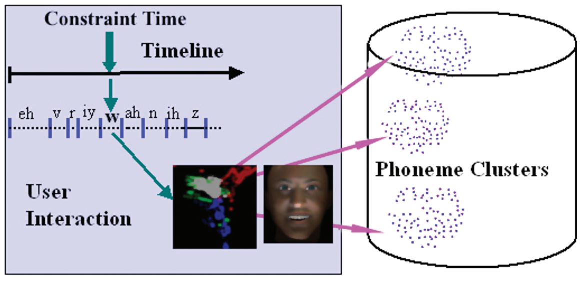


**Figure 8:** *Illustration of how to specify a motion-node constraint via the phoneme–Isomap interface. When users want to specify a specific motion node for expressing a particular phoneme utterance, its corresponding phoneme Isomap is automatically loaded. Then, users can interact with the system to specify a motion-node constraint for this constrained phoneme.*

### 6.1 Specify motion-node constraints

Users interactively browse phoneme Isomap images to specify motion-node constraints and tie them to a specific phoneme utterance’s expression. We refer to this time as *a constrained time* and its corresponding phoneme as *a constrained phoneme*. Phoneme timing is enclosed in the preprocessed phrase (phoneme) transcript, so a phoneme Isomap image is automatically loaded once a constrained phoneme is selected (Figure 8).

To guide users in identifying and selecting proper motion nodes, our system automatically highlights recommended motion nodes and their picking points. Assuming a motion node path $o_1, o_2, \ldots, o_k$ is obtained by the automatic motion-path search algorithm (the follow-up Section 6.2 will detail this algorithm), users want to specify a motion-node constraint for a constrained time $T_c$ (assume its corresponding constrained phoneme is $PHO_c$ and its motion-frame at $T_c$ is $FRM_c$, called *current selected frame*). The constrained time $T_c$ is first divided by the duration of the constrained phoneme $PHO_c$ to calculate its relative time $t_c (0 \leq t_c \leq 1)$. Then, for

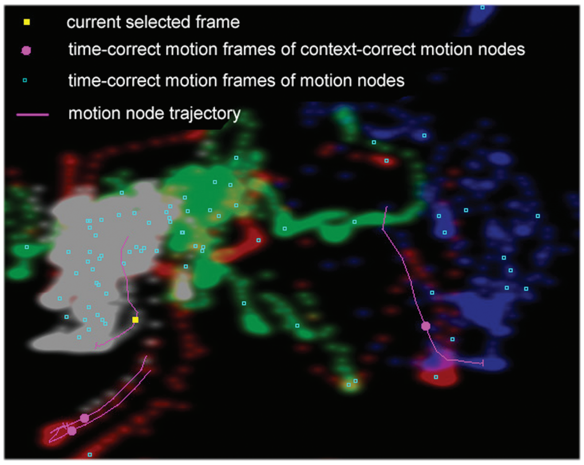

current selected frame
time-correct motion frames of context-correct motion nodes
time-correct motion frames of motion nodes
motion node trajectory

**Figure 9:** *A snapshot of phoneme–Isomap highlights for specifying motion-node constraints.*

each motion node in the phoneme cluster, this system interpolates and highlights one motion frame whose *relative time* attribute is the current relative time $T_c$. We refer to these motion frames as *time-correct motion frames*.

As mentioned in Section 3, the specific triphone context of each motion node was also retained. By matching the triphone context of $PHO_c$ with that of other existing motion nodes in the phoneme cluster (of $PHO_c$), this system identifies and highlights the motion nodes in the phoneme cluster that have the same triphone context as $PHO_c$ (termed *context-correct motion nodes*). For example, in Figure 8, the current constrained phoneme is $PHO_c = /w/$, and its triphone context is [/ iy /, / w /, / ah /], so the system will identify the motion nodes of the / w / phoneme cluster that have the same triphone context [/ iy /, / w /, / ah /] as the context-correct motion nodes. In this way, by picking their representative time-correct motion frames, users can choose one of those motion nodes as a motion-node constraint for $PHO_c$. This motion node constraint is imposed per phoneme utterance, in other words, if one specific phoneme appears multiple times in a phoneme input sequence, users can specify different motion-node constraints for them. Figure 9 shows a snapshot of phoneme Isomap highlights for specifying motion-node constraints. The background phoneme Isomap image is always the same for a specific phoneme, but these highlighting symbols (Figure 9) are related to current relative time $t_c$ and current triphone context. These markers are typically changed over time (even for the same phoneme).

## 6.2. Optimal motion node search

Now we describe how our algorithm searches for optimal combinations of motion nodes from the constructed facial motion node database while satisfying various constraints. We can formalize this motion-node path search problem as follows: given a novel phoneme sequence input $\Psi = (P_1, P_2, \cdots, P_T)$ and its emotion modifiers $\Theta = (E_i, E_2, \cdots, E_T)$, $E_i$ can only be one of four possible values: neutral, anger, sadness and happiness, and optional motion-node constraints $\Phi = (C_{t_1} = o_{i_1}, C_{t_2} = o_{i_2}, \cdots, C_{t_k} = o_{i_k}, t_i \neq t_j)$, we want to search for a best-matched motion-node path $\Gamma^* = (o^*_{\rho_1}, o^*_{\rho_2}, \cdots, o^*_{\rho_T})$ that minimizes a cost function $COST(\Psi, \Theta, \Phi, \Gamma)$. Here, $o_i$ represents a motion node with index $i$.

The cost function $COST(\Psi, \Theta, \Phi, \Gamma)$ is the accumulated summation of a *Transition Cost* $TC(o_{\rho_i}, o_{\rho_{i+1}})$, an *Observation Cost* $OC(P_i, o_{\rho_i})$, an *Emotion Mismatch Penalty* $EMP(E_i, o_{\rho_i})$, and a *Blocking Penalty* $B(i, o_{\rho_i})$, as described in Equation (4).

$$COST(\Psi, \Theta, \Phi, \Gamma) = \sum_{i=1}^{T-1} TC(o_{\rho_i}, o_{\rho_{i+1}}) + \sum_{i=1}^{T} (OC(P_i, o_{\rho_i}) + EMP(E_i, o_{\rho_i}) + B(i, o_{\rho_i})). \quad (4)$$

Here, $TC(o_{\rho_i}, o_{\rho_{i+1}})$ represents the smoothness of the transition from a motion node $o_{\rho_i}$ to the other motion node $o_{\rho_{i+1}}$. The rules for computing $TC(o_{\rho_i}, o_{\rho_{i+1}})$ are shown in Table 1. $OC(P_i, o_{\rho_i})$ measures the goodness of a motion node $o_{\rho_i}$ for expressing a given phoneme $P_i$. The rules for

**Table 1:** *Rules of computing the transition cost* $TC(o_{\rho_i}, o_{\rho_{i+1}})$

| Rule conditions | Transition cost (TC) |
|---|---|
| **Rule 1**: $pre(o_{\rho_{i+1}}) = o_{\rho_i}$ | 0 |
| **Rule 2**: viseme $(o_{\rho_i}) =$ viseme $(pre(o_{\rho_{i+1}}))$ | $\alpha_2.DSC(o_{\rho_i}, pre(o_{\rho_{i+1}})) + PVC(o_{\rho_i}, o_{\rho_{i+1}})$ |
| **Rule 3**: viseme $(o_{\rho_i}) \neq$ viseme$(pre(o_{\rho_{i+1}}))$ | $\alpha_2.DSC(o_{\rho_i}, pre(o_{\rho_{i+1}})) + PVC(o_{\rho_i}, o_{\rho_{i+1}}) + PNT$ |
| **Rule 4**: $pre(o_{\rho_{i+1}}) = NIL$ | $\alpha_1.PNT$ |

Here, the priorities of these rules are: **Rule 1** > **Rule 2** > **Rule 3** > **Rule 4**. If any rule with a higher priority can be applied, then the lower rules will be ignored.

**Table 2:** *Rules of computing the observation cost* $OC(P_i, o_{\rho_i})$

| Rule conditions | Observation cost |
|---|---|
| **Rule 1**: $P_i = pho(o_{\rho_i})$ or $P_i = /pau/$ | 0 |
| **Rule 2**: viseme$(P_i) =$ viseme$(o_{\rho_i})$ | $\alpha_4.\alpha_5.PNT$ |
| **Rule 3**: the remaining cases | $\alpha_5.PNT$ |

Here, the priorities of these rules are: **Rule 1 > Rule 2 > Rule 3**. If any rule with a higher priority can be applied, then the lower rules will be ignored.

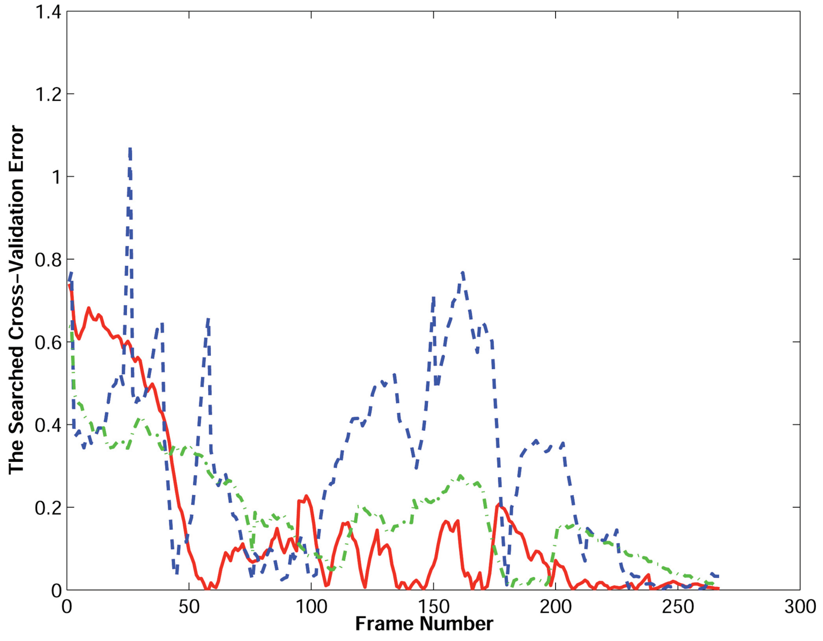


**Figure 10:** *The searched cross-validation error-per-frame for three chosen sentences. Here, a red solid curve for a neutral sentence,a blue dash curve for an angry sentence, and a green dash-dot curve for a sad sentence.*

computing $OC(P_i, o_{\rho_i})$ is shown in Table 2. $EMP(E_i, o_{\rho_i})$ is defined in Equation (5). The blocking cost item $B(i, o_{\rho_i})$ will be described later. In this section, $\langle \alpha_i \rangle_{i=1}^{6}$ are pre-defined constants.

$$EMP(E_i, o_{\rho_i}) = \begin{cases} 0 & \text{if } E_i = emotion(o_{\rho_i}) \\ \alpha_6.PNT & \text{otherwise.} \end{cases} \quad (5)$$

The explanation of the above Equation (5) is that, if the emotion label of a motion node $o_{\rho_i}$ is the same as the specified emotion modifier $E_i$, we set the emotion mismatch penalty to zero, otherwise it is set to a penalty $\alpha_6$.*PNT*.*PNT* is a pre-defined constant.

The explanation of the rules of computing the transition cost $TC(o_{\rho_i}, o_{\rho_{i+1}})$ (shown in Table 1) is: if $o_{\rho_i}$ is the captured predecessor motion node of $o_{\rho_{i+1}}$ (i.e. $pre(o_{\rho_{i+1}}) = o_{\rho_i}$), their transition cost is set to zero (i.e. a perfect transition). If the phoneme of $pre(o_{\rho_{i+1}})$ exists and belongs to the same viseme category of that of $o_{\rho_i}$, then the cost value is the weighted sum of a direct smoothing cost $DSC(o_{\rho_i}, pre(o_{\rho_{i+1}}))$ and a position velocity cost $PVC(o_{\rho_i}, o_{\rho_{i+1}})$. Equations (6) to (9) describe how to compute $DSC(o_{\rho_i}, pre(o_{\rho_{i+1}}))$ and $PVC(o_{\rho_i}, o_{\rho_{i+1}})$. If these two motion nodes do not share the same viseme category, then a penalty value *PNT* is added. If $pre(o_{\rho_{i+1}})$ does not exist, we set a modulated penalty value $\alpha_1 * PNT$. For the definition of viseme categories used in this work, please refer to the *Appendix A*.

The explanation of the rules of computing the transition cost $OC(P_i, o_{\rho_i})$ (shown in Table 2) is : if the phoneme of $o_{\rho_i}$ is the expected $P_i$ or $P_i$ is the silence phoneme/pau/, this cost is set to zero. If they are in the same viseme category, then it is set to a discounted penalty value $\alpha_4.\alpha_5.PNT$ and $0 < \alpha_4 < 1$, otherwise, the cost is a penalty value $\alpha_5.PNT$.

$$DSC(o_{\rho_i}, pre(o_{\rho_{i+1}})) = \int Blend(warp(o_{\rho_i}), pre(o_{\rho_{i+1}}))''dt \quad (6)$$

$$PVC(o_{\rho_i}, o_{\rho_{i+1}}) = \alpha_3.PGap(o_{\rho_i}, o_{\rho_{i+1}}) + VGap(o_{\rho_i}, o_{\rho_{i+1}}) \quad (7)$$

$$PGap(o_{\rho_i}, o_{\rho_{i+1}}) = \sum_{k=1}^{MrkNum} \sqrt{\|FP_k(o_{\rho_{i+1}}) - LP_k(o_{\rho_i})\|^2} \quad (8)$$

$$VGap(o_{\rho_i}, o_{\rho_{i+1}}) = \sum_{k=1}^{MrkNum} \sqrt{\|FV_k(o_{\rho_{i+1}}) - LV_k(o_{\rho_i})\|^2}. \quad (9)$$

The above Equations (6) to (9) define the Direct Smoothing Cost $DSC(o_{\rho_i}, pre(o_{\rho_{i+1}}))$ and position velocity cost $PVC(o_{\rho_i}, o_{\rho_{i+1}})$. To compute the direct smoothing cost $DSC(o_{\rho_i}, pre(o_{\rho_{i+1}}))$, first time-warp $o_{\rho_i}$ to make it align with $pre(o_{\rho_{i+1}})$ frame by frame, then do a linear blend on the time-warped motion $warp(o_{\rho_i})$ and $pre(o_{\rho_{i+1}})$, finally, compute the integral of the second derivative of the blended motion as the direct smoothing cost. $PVC(o_{\rho_i}, o_{\rho_{i+1}})$ is the weighted sum of position gap *PGap* and velocity gap *VGap* between the end of $o_{\rho_i}$ and the start of $o_{\rho_{i+1}}$.

In Equations (8) and (9), *MrkNum* is the number of the used markers, $FP_k(o_i)$ represents the 3D position of the $k^{th}$ marker of the first frame of $o_i$ motion node, and $LP_k(o_i)$ represents the 3D position of the $k^{th}$ marker of the last frame of $o_i$ motion node. $FV_k(.)$ and $LV_k(.)$ are defined in a similar way, but for 3D velocity.

Now we describe how the specified motion-node constraints $\Phi = (C_{t_1} = o_{i_1}, C_{t_2} = o_{i_2}, \cdots, C_{t_k} = o_{i_k}, t_i \neq t_j)$ affect the cost function to guarantee that the searched motion-node path passes through the specified motion nodes at specified times. The constraints block the chances of other motion nodes (except the specified ones) for representing constrained phoneme utterances. An additional cost term $B(t, o_j)$ is introduced for this purpose (Equation 10):

$$B(t, o_j) = \begin{cases} 0 & \text{if } \exists m, t_m = t \text{ and } j = i_m \\ Huge\ Penalty & \text{otherwise.} \end{cases} \quad (10)$$

Based on the above cost definitions, we use the dynamic programming algorithm to search for the best-matched motion node sequence in the database. Assume that there are total $N$ motion nodes in the processed motion node database and the length of a new phoneme transcript input is $T$. This expressive speech animation synthesis algorithm can be briefly summarized in Algorithm 1.

The time complexity of the above search algorithm is $\Theta(N^2.T)$, here $N$ is the number of motion nodes in the database and $T$ is the length of input phonemes. Note that in the above search algorithm (Algorithm 1 and Equations (4) to (10), constant parameters $\langle \alpha_i \rangle_{i=1}^{6}$ are used to balance the weights of different costs. In this work, the cross-validation technique [HRF01] was used to experimentally determine these parameter values (refer to Section 6.4 for more details).

Given the optimal motion-node path $\Gamma^*$, we concatenate its motion nodes by smoothing their neighbouring boundaries and transforming motion nodes from their retained PCA space to markers' 3D space (Equation 11). Finally, we transfer the synthesized marker motion sequence to specific 3D face models:

$$MrkMotion = MeanMotion + EigMx.PcaCoef. \quad (11)$$

Here *MeanMotion* represents the mean motion vector computed in the previous PCA reduction procedure (Section 3), *EigMx* represents the retained eigen-vector matrix composed of the largest 25 eigen-vectors, and *PcaCoef* represents a 25-dimensional PCA coefficient vector.

Algorithm 1 *Expressive Speech Motion Synthesis*

```
Input: OC[$P_1$ ...$P_T$, $o_1$ ...$o_N$], observation cost function.
Input: EMP[$E_1$ ...$E_T$, $o_1$ ...$o_N$], emotion mismatch penalty
    function.
Input: TC[$o_1$ ...$o_N$, $o_1$ ...$o_N$], transition cost function.
Input: B[1...T, $o_1$ ...$o_N$], blocking penalty.
Input: N, size of the facial motion node database; T, length of
    input phoneme sequence.
Output: Motion, synthesized expressive facial motion sequence.
 1: for $i = 1$ to $N$
 2:   $\varphi_1(i) = OC(P_1, o_i) + EMP(E_1, o_i)$
 3: end for
 4: for $t = 2$ to $T$
 5:   for $j = 1$ to $N$
 6:     $\varphi_t(j) = \min_i\{\varphi_{t-1}(i) + TC(o_i, o_j) + OC(P_t, o_j) + EMP(E_t, o_j) + B(t, o_j)\}$
 7:     $\chi_t(j) = \arg\min_i\{\varphi_{t-1}(i) + TC(o_i, o_j) + OC(P_t, o_j) + EMP(E_t, o_j) + B(t, o_j)\}$
 8:   end for
 9: end for
10: $COST^* = \min_i \{\varphi_T(i)\}$
11: $\rho_T^* = \arg\min_i\{\varphi_T(i)\}$
12: for $t = T - 1$ to 1
13:   $\rho_t^* = \chi_{t+1}(\rho_{t+1}^*)$
14: end for
15: PcaSeq = ConcatenateAndSmooth($o_{\rho_1}^*, o_{\rho_2}^*, \cdots, o_{\rho_T}^*$)
16: Motion = PcaTransformBack(PcaSeq)
```

## 6.3. GPU-accelerated motion node search

In the above *Expressive Speech Motion Synthesis* algorithm (Algorithm 1), its dynamic programming part consumes the majority of the computing time of Algorithm 1. As such, we accelerate the dynamic programming part by implementing it on GPU as follows. We use fragment shaders written in NVIDIA's Cg programming language to calculate cost values in Algorithm 1. All data fields (including viseme, phoneme and emotion) are stored as RGBA textures in GPU. In this implementation, we use five data textures, as listed in Table 3.

The above data textures are updated as follows: first render data to a pixel buffer using a fragment program that performs the needed dynamic programming-based search operations, and then copy the content of the pixel buffer to the final destination of the data texture. Note that we fetch *DistanceMap* each time before we calculate the cost value. Hence, actually we do not need to store all distance data in video memory at one time.

## 6.4. Cross-validation for cost trade-off

As mentioned in Section 6.2, parameters $\langle \alpha_i \rangle_{i=1}^{6}$ balance the weights of costs from different sources, which is critical to the success of this expressive speech animation synthesis algorithm. Additional 24 sentences (each emotion has six), not used in the training database, are used to determine these

**Table 3:** *Data texture packing*

| Texture | Content |
|---|---|
| CostBuffMap | [cost,index,-,-] |
| CostValueMap | [cost] |
| RetraceIndexMap | [index] |
| MotionNodeMap | [seq, phoneme, viseme, emtion] |
| PrevCostBufferMap | [preCost, preIndex, -, -] |

optimal parameters by cross-validation [HRF01]. A metric (Equations 12 and 13) is introduced to measure the error between ground-truth motion capture data and synthesized marker motion. We used gradient-descent methods [Pie86] to search for optimal parameter values. In case the search process may become stuck at a local minimum, we ran the gradient-descent process hundreds of times and each time it started at a random place. Finally, we picked a certain combination of parameters that leads to the minimum value among all searched results as the optimal parameter setting. Figure 10 shows the searched cross-validation error-per-frame (Equation 13) for three chosen sentences.

$$TotalErr = \sum_{i=1}^{K} Err_i \tag{12}$$

$$Err_i = \frac{\sum_{j=1}^{N_i} \sum_{k=1}^{MrkNum} \| SYN_i^{j,k} - ORI_i^{j,k} \|^2}{(N_i * MrkNum)}. \tag{13}$$

Here, $K$ is the number of cross-validation sentences, $MrkNum$ is the number of the used markers, $N_i$ is the total number of frames of the $i$th sentence, $SYN_i^{j,k}$ is the 3D position of the $k$th synthetic marker of the $j$th frame of the $i$th sentence, and $ORI_i^{j,k}$ is for the motion capture ground-truth.

## 7. 3D Face Deformation and Motion Smoothing

In this section, we describe how to transfer the synthesized marker motion sequence to a static 3D face model, and how to use smoothing and resampling techniques to concatenate searched motion nodes.

### 7.1. 3D face deformation

After facial marker motions are synthesized, we need to transfer these motions to a specific 3D face model. Figure 11 illustrates the used 3D face model.

Although radial basis functions (RBF) have been demonstrated successful applications in facial animation deformation [DCFN06, LCF00, NN01, PHL*98], we extended the feature point based mesh deformation approach proposed by Kshirsagar *et al.* [KGT01] for this deformation due to its efficiency.

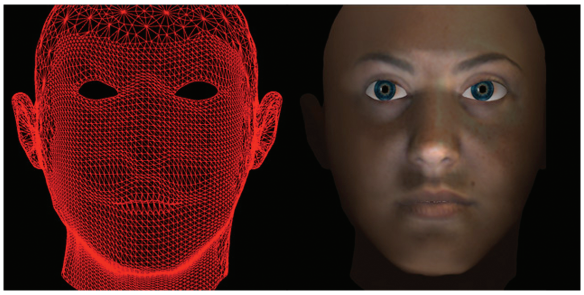


**Figure 11:** *Illustration of the 3D face model used in this work. The left panel is a wireframe representation and the right panel is a textured rendering (with eyeball and teeth).*

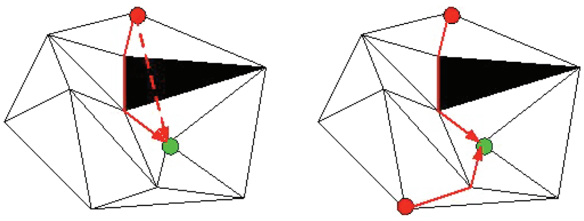


**Figure 12:** *The left panel compares mesh distance (solid red line) with the direct Euclidean distance (dotted red line) from the source marker/vertex (red spot) to the target vertex (green spot). The black area represents a hole in the face geometry. The right panel shows how the motion of one vertex is calculated based on the propagated motion from several neighbouring markers.*

First, for each facial marker (corresponding to a picked vertex on the face model), 'mesh distances' from this facial marker (source) to other vertices (targets) on the face model were computed. Here 'mesh distance' is the summation of distances of the shortest path along the mesh surface, not the direct Euclidean distance from the source vertex (marker) to the target vertex. The smaller the mesh distance is, the more the target vertex is affected by the source marker. The underlying reasons include: (i) There are some holes (e.g. mouth and eyes area) in the face model, and direct Euclidean distance would lead to motion interferences. For example, the direct Euclidean distance between a lower lip marker and some vertices on the upper lip may be small, but the fact is that the lower lip marker has little effect on the motion of the upper lip. (ii) 'Mesh distances' between vertices can better approximate face surface geometry than direct Euclidean distances, and hence it is more proper for measuring facial motion propagation. It should be noted that the accuracy of 'mesh distances' depends on the density and quality of face meshes, and more accurate 'mesh distances' can be calculated from denser face meshes. Figure 12 illustrates mesh distance.

Then, if the mesh distance from a facial marker to a target vertex is less than a threshold, we say *this target vertex is affected by the facial marker*. The contributing weights of the facial markers to affected vertices are computed based on a normal distribution centred at the facial marker place (their mesh distance as a parameter). The right panel of Figure 12 shows how multiple markers may affect a certain vertex on the face model. In practice, for each vertex of the face model, we precomputed and stored the contributing weight of each facial marker (feature point). Then, after the precomputed contributing weights are loaded, our system can deform the face on-the-fly at an interactive rate given any marker motion frame (Equations 14 to 15).

$$w_{k,i} = C.\exp^{-MeshDist(k,i)^2/(2.\sigma^2)} \tag{14}$$

$$VtxMotion_i = \frac{\sum_{k=1}^{MrkNum} w_{k,i}.MrkMotion_k}{\sum_{k}^{MrkNum} w_{k,i}}. \tag{15}$$

Here, $C$ is a constant value, $VtxMotion_i$ is the motion of the $i$th vertex of the face model, and $MrkMotion_k$ is the motion of the $k$th marker, and $w_{k,i}$ is the weight of the $k$th marker contributing to the motion of the $i$th vertex.

In the above Equation (14), we experimentally set the value of $\sigma$ to 1.5. It should be noted that the major difference between our approach and Kshirsagar *et al.*'s work [KGT01] is that we employ a different algorithm to compute weights (Equation 14).

### 7.2. Smoothing and resampling

Undesired changes may exist between concatenated motion nodes, so we need to use smoothing techniques to smooth transitions. In this work, we extend the trajectory-smoothing technique [EGP02, MCP*04] for smoothing operations. The smoothing operation from a motion node $o_j$ to another motion node $o_k$ can be described as follows: assuming smoothing window size is $2s$, $f_1$ to $f_s$ are the ending $s$ frames of the motion node $o_j$, $f_{s+1}$ to $f_{2s}$ are the starting $s$ frames of next motion node $o_k$, and $f_i = f(t_i)$, $1 \le i \le 2s$, we want to find a smooth curve *g(t)* to best fit $f(t_i)$ by minimizing the following objective function:

$$\sum_{i=1}^{s} W_j.(g_i - f_i)^2 + \sum_{i=s+1}^{2s} W_k.(g_i - f_i)^2 + \int_{t_1}^{t_{2s}} g''(t)dt \tag{16}$$

In the above Equation (16), the first and second items are used to measure how close *g(t)* is to $f(t_i)$, and the third term is used to measure the smoothness of *g(t)*. In other words, the expected smooth curve *g(t)* should be close to $f(t_i)$ while as smooth as possible. $s$ is experimentally set to 3 in this work. $W_j$ (or $W_k$) is a pre-assigned weight factor for the phoneme of the motion node $o_j$ (or $o_k$). Because this value depends on the specific phoneme (of one motion node), in this work, we categorize all phonemes into three groups (similar to the phoneme categories proposed in Pelachaud's work [Pel91]): high visible phonemes (e.g. /p/, /b/, /m/, etc.) for a high weight factor (=2), low visible phonemes (e.g., /k/, /g/, /ch/, etc.) for a low weight factor (=0.5), and intermediate visible phonemes for a middle weight factor (=1.0). It should be noted that these weight values are experimentally determined, and they might be specific to this database. Table 4 shows weight factors of phonemes used in the smoothing process. We only apply this smoothing technique to two concatenated motion nodes without a captured predecessor/successor relation.

**Table 4:** *Weight factors of different phonemes used in the smoothing techniques*

| Phonemes | Weight factor |
|---|---|
| /p /, /b/, /m/, /f/, /v/, /w/ | High = 2 |
| /k /, /g/, /ch/, /ng/, /r/ | Low = 0.5 |
| Others | Intermediate = 1 |

As mentioned in Section 3, motion nodes for the silence time (the/pau/phoneme) were discarded when constructing the facial motion node database. When computing the observation cost for the/pau/phoneme time (Table 2), as long as $P_i = /pau/$, we simply set the observation cost to zero. In other words, any motion node is perfect for expressing the silence time during the motion node search process (Section 6.2). After motion nodes are concatenated and smoothed, we need to post-process these synthesized frames corresponding to the silence time: first identify these silence-time frames based on the input phoneme transcript and then regenerate these frames by performing a linear interpolation on the boundary of non-silence frames. It should be noted that besides linear interpolation, other interpolation techniques including cubic spline and cosine interpolation can be used for this interpolation process too. Here, we choose linear interpolation due to its efficiency. Figure 13 illustrates this step.

During the post-processing stage, it is necessary to resample motion frames. When motion nodes are concatenated, the number of frames of the motion node may not exactly match the duration of the input phoneme. We use the time-warping technique to resample the searched motion nodes to obtain the desired number of motion frames. This resampling is still done at 120 Hz (the same as the original motion capture rate). Although synthesized marker motion frames are 120 frames/second, the resulting animations are often at an ordinary animation rate of 30 frames/second. Thus, before we transfer the synthesized marker motion to a 3D face model, we down-sample these motion frames to the ordinary animation rate.

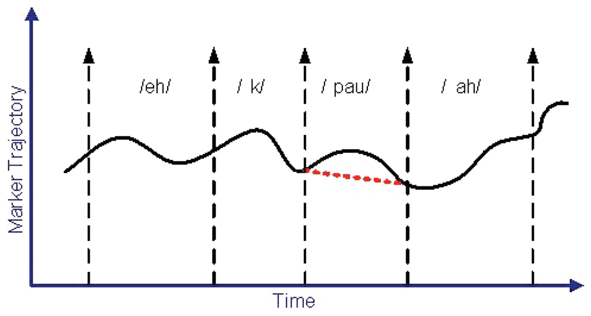
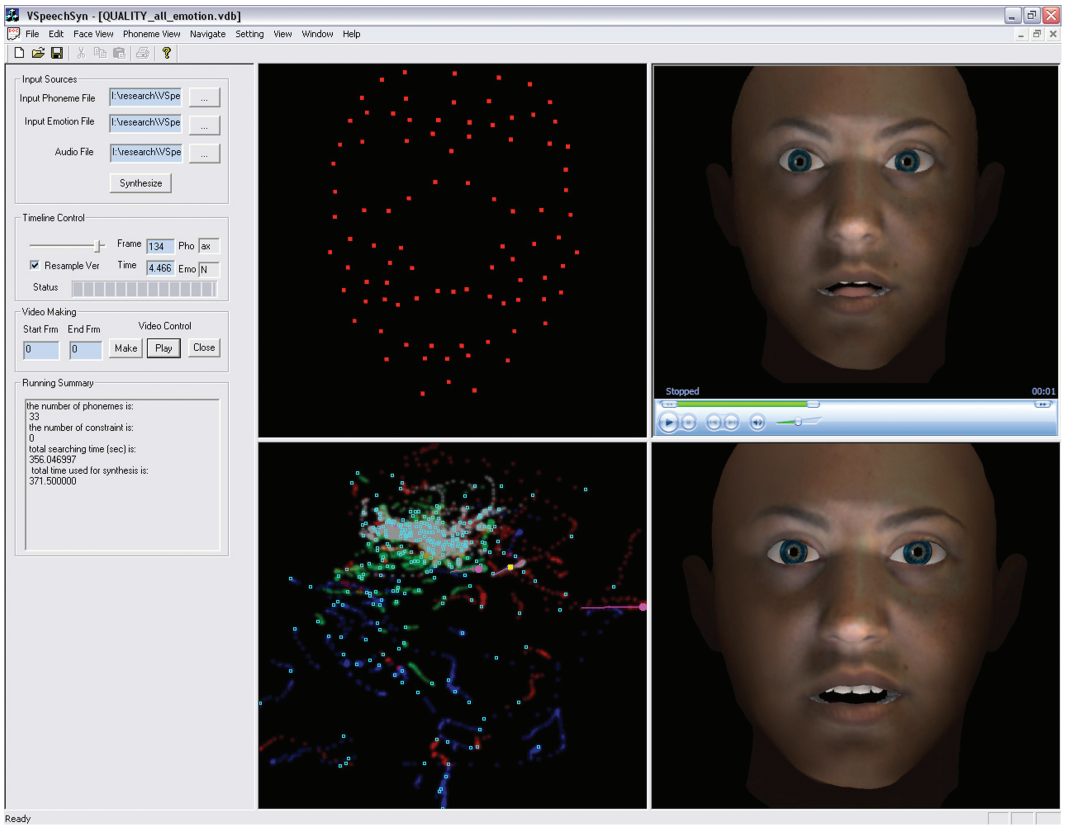


**Figure 13:** *Illustration of the post-processing step for the silence time (the /pau/ phoneme time). Here, red dot line represents regenerated marker trajectory.*

## 8. Results and Evaluations

We developed this system on the MS Windows XP system. Figure 14 shows a snapshot of the running system. The left is a basic control panel, and the right panel encloses four working windows: a synthesized motion window (top-left), a video playback window (top-right), a phoneme–Isomap interaction window (bottom-left) and a face preview window (bottom-right). The synthesized motion window and the face preview windows can switch among several display modes, including marker-drawing mode and deformed 3D face mode. In the basic control panel, users can input a novel speech (WAV format) and its aligned phoneme transcript file (text format), and an emotion specification (modifier) file (text format), then the system automatically synthesizes its corresponding expressive facial animation (shown in the synthesized motion window). Once the facial motion is synthesized, users can interactively browse every frame and play back the animation in the synthesized motion window (top-left in Figure 14). Additionally, the system can automatically compose an AVI video (audio-synchronized), which user can play back immediately in the video playback window (top-right in Figure 14) to check the final result.

On the user interaction side, users can edit the facial motion database and impose motion-node constraints via the phoneme–Isomap interaction window (bottom-left in Figure 14) and the face preview window (bottom-right in



Input Phoneme File: I:\research\VSpe
Input Emotion File: I:\research\VSpe
Audio File: I:\research\VSpe
Frame: 134 Pho: ax
[✓] Resample Ver Time: 4.466 Emo: N
Status
Start Frm: 0 End Frm: 0

```
the number of phonemes is:
33
the number of constraint is:
0
total searching time (sec) is:
356.046997
 total time used for synthesis is:
371.500000
```

**Figure 14:** *A snapshot of this running system. The left is a basic control panel, and the right panel encloses four working windows: a synthesized motion window (top-left), a video playback window (top-right), a phoneme–Isomap interaction window (bottom-left), and a face preview window (bottom-right).*

**Table 5:** *Running time of synthesis of some example phrases*

| Phrases (Number of phonemes) | Running time without GPU-accelerated (second) | Running time with GPU-accelerated (second) | Speedup Ratio |
|---|---|---|---|
| 'I know you meant it' (14) | 221.391 | 50.828 | 4.356 |
| 'And so you just abandoned them?' (24) | 333.171 | 90.891 | 3.666 |
| 'Please go on, because Jeff's father has no idea of how things became so horrible'. (53) | 815.719 | 232.078 | 3.515 |
| 'It is a fact that long words are difficult to articulate unless you concentrate' (63) | 942.609 | 256.484 | 3.675 |

Here, the computer used is a PC (Windows XP, 1GHz Memory, Athlon 64X2 Dual-Core, NVIDIA GeForce 6150 LE graphics card). The used motion node database encloses 5556 motion nodes.

Figure 14). If a point in the phoneme–Isomap interaction window is picked, the face preview window will show the deformed 3D face (or corresponding facial marker configuration) interactively.

We conducted a running time analysis on this system (both GPU-accelerated and non-GPU accelerated). The computer used is a PC (Windows XP, 1GHz Memory, Athlon 64X2 Dual-Core, NVIDIA GeForce 6150 LE graphics card). The used motion node database encloses 5556 motion nodes. Table 5 encloses the running time of some example inputs. As mentioned in Section 6.2, the motion node searching part (the most time-consuming part of this system) has a time complexity of $\Theta(N^2.T)$ that is linear to the length of input phonemes (assuming $N$ is a fixed value for a specific database). The experimental computing time enclosed in the Table 5 is approximately matched with this theoretical analysis. In addition, comparing to the non-GPU implementation, our GPU-accelerated system decreases the running time significantly.

We also compared the synthesized expressive facial motion with ground-truth captured motion. Twelve additional sentences were exclusively used for test comparisons. One of these sentences was 'Please go on, because Jeff's father has no idea of how the things became so horrible'. We chose a lower lip marker (#79 marker) in a speech-active area for the comparisons (the specific location of #79 marker can be found at Figure 2). We plotted a part of the synthesized sequence and ground truth motion for marker trajectory and velocity comparisons. Figure 15 is for trajectory comparisons and Figure 16 is for marker velocity comparisons. We found that the trajectories of the synthesized motions are quite close to the actual motions captured from the actress, but their velocities deviated more. Notice that in this ground-truth comparison, the synthesized motions for these comparisons (Figures 15 to 16) were automatically generated *without any manual intervention* (i.e. without the use of motion-node constraints).

We also synthesized numerous expressive speech animations using novel recorded and archival speech. In most of cases, without any manual intervention our system can generate facial animations with reasonable quality. However, to improve their animation quality, we typically specify motion constraints two to four times per sentence, which takes several minutes.

## 9. Discussion and Conclusions

We present a data-driven expressive speech animation synthesis system with intuitive phoneme-level controls. Users can control facial motion synthesis process by specifying emotion modifiers and expressions for certain phoneme utterances via 2D expressive phoneme Isomaps introduced in this work. This system employs a constrained dynamic programming algorithm that satisfies 'hard constraints' (motion-node constraints) and 'soft constraints' (emotion modifiers). Objective comparisons between synthesized facial motion and ground truths (captured facial motion), and novel synthesis experiments, demonstrate that this system is effective for generating realistic expressive speech animations based on expressive speech input.

To build intuitive controls for expressive facial animation synthesis, this system introduces the Isomap framework [TSL00] for generating intuitive low-dimensional manifolds for each phoneme cluster. The advantage of the Isomap (over PCA, for example) is that it leads to a better projection of motion frames with different emotions, and it makes browsing and managing expressive facial motion sequences (and frames) more intuitive and convenient. An interactive and intuitive way of browsing and selecting among the large number of phoneme variations is itself a challenging problem for facial animation research.

As this is a new approach to facial animation synthesis and control, several issues remain further investigations. The quality of novel facial motion synthesis depends on constructing a large facial motion database with accurate motion and phoneme alignment. Building this database takes care and time; integrated tools could improve this process immensely. A large amount of expressive facial motion data

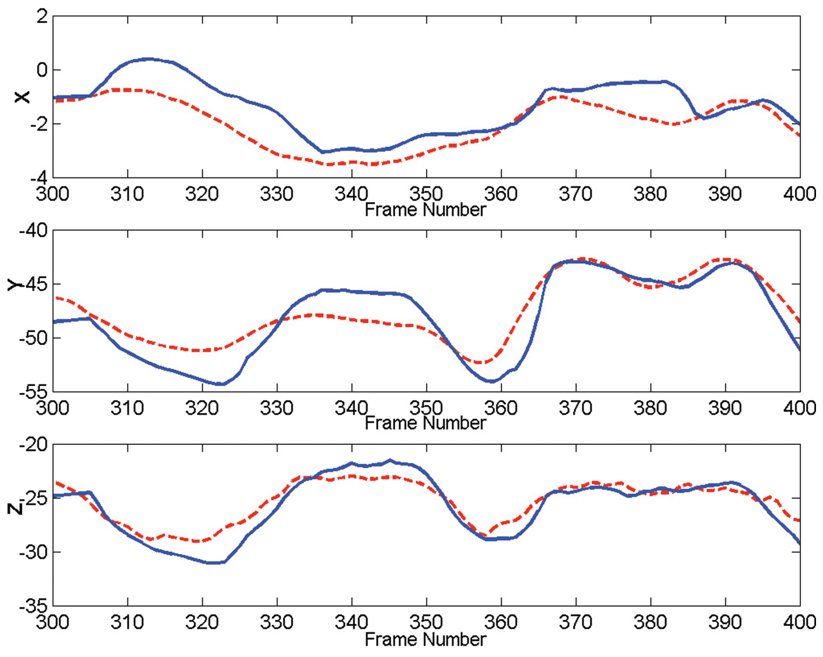


**Figure 15:** *Comparison of the trajectory of the lip marker (#79 marker). The dashed line is the ground truth trajectory and the solid line is the synthesized trajectory.*

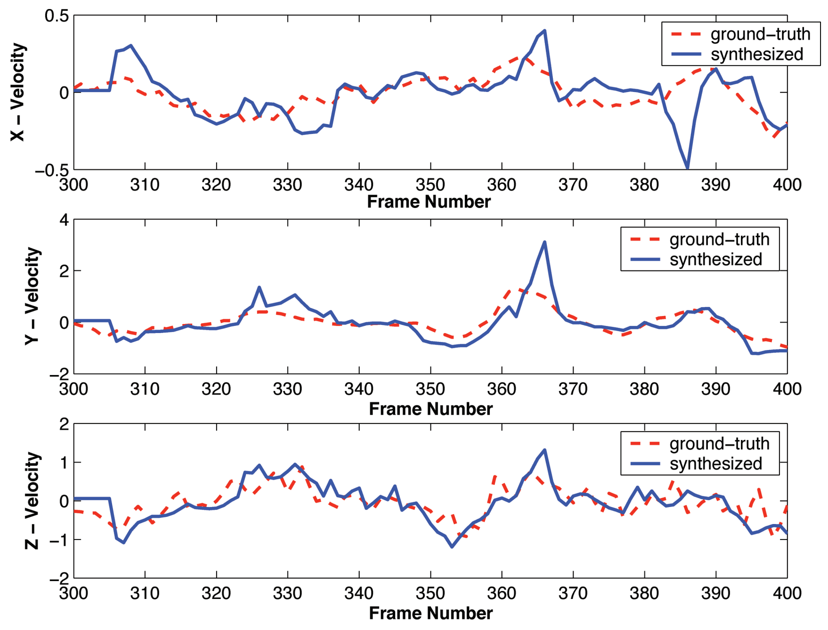


**Figure 16:** *Comparison of the velocity of the lip marker (#79 marker). The dashed line is the ground truth trajectory and the solid line is the synthesized trajectory.*

are needed to construct the database. The larger the captured facial motion database is, the better the synthesis results we expect from this system. However, it is difficult to anticipate in advance how much data are needed to generate realistic facial animations, which is one of the unresolved issues in many data-driven systems. Further research on the trade-off between synthesis quality and the size of captured facial motion database is certainly a priority. The current system cannot be used for real-time applications. Optimizations could further improve its efficiency by reducing the size of the facial motion database through clustering methods.

We are also aware that subjective evaluation would be helpful to quantify and improve this system. We conducted a small-scale user evaluation on this system and its phoneme–Isomap interface, and most of participants are undergraduate/graduate students working in the Lab. Initial feedback from them is quite positive. We plan to design a rigorous in-depth study to look into this issue in the future. In addition, emotion intensity control that is absent in the current system is another good direction to go for future research.

The motions of the silence phoneme (the /pau/ phoneme in the *Festival* system) are not modelled. This phoneme and other non-speaking behaviours (e.g. yawning ) need to be represented as motion nodes to allow more flexibility and personified realism in the future work. Lastly, there are more open questions, such as whether combining the speaking styles of different actors into one facial motion database would result in providing a greater range of motions and expressions, or if such a combination would muddle the motion-frame sequencing and expressiveness, or whether exploiting different weights for markers to guide the coherence of perceptual saliency could improve results.

## Acknowledgments

We would like to thank Xiaohan Ma for his GPU programming, Yizhou Yu for thoughtful suggestions, Pamela Fox for face model preparation, Joy Nash, J.P. Lewis, Murtaza Bulut and Carlos Busso for facial motion data capture and processing, and Douglas Fidaleo and the USC Computer Vision Lab for building the coarse 3D face model.

## Appendix A: Viseme Categories Used in This Work

In order to generate visual speech animations, researchers often create a phoneme-viseme mapping: if multiple phonemes (the basic unit of speech in the acoustic domain) have simliar mouth shapes when they are pronounced, these phonemes are typically mapped to a common visual representation (termed as a viseme). In this work, the phoneme-viseme scheme given in Table A1 defines a total of 15 viseme categories.

**Table A1:** *Scheme of grouping phonemes into the 15 visemes used in this work*

| /pau/ | /r/ | /k/, /g/, /ng/ |
|---|---|---|
| /ae/, /ax/, /ah/, /aa/ | /f/, /v/ | /ch/, /sh/, /jh/ |
| /ao/, /y/, /iy/, /ih/, /ay/, /aw/ | /ow/, /oy/ | /n/, /d/, /t/, /l/ |
| /ey/, /eh/, /el/, /em/, /en/, /er/ | /th/, /dh/ | /s/, /z/, /zh/ |
| /b/, /p/, /m/ | /hh/ | /w/, /uw/, /uh/ |

## References

[AF02] Arikan O., Forsyth D. A.: Interactive motion generation from examples. *ACM Transactions on Graphics* 21 (2002), 483–490.

[BBPV03] Blanz V., Basso C., Poggio T., Vetter T.: Reanimating faces in images and video. *Computer Graphics Forum 22*, 3 (2003), 641–650.

[BCS97] Bregler C., Covell M., Slaney M.: Video rewrite: Driving visual speech with audio. *Proceedings of ACM SIGGRAPH'97* (1997), pp. 353–360.

[BDNN05] Busso C., Deng Z., Neumann U., Narayanan S.: Natural head motion synthesis driven by acoustic prosody features. *Computer Animation and Virtual Worlds 16*, 3-4 (2005), 283–290.

[BH00] Brand M., Hertzmann A.: Style machines. In *Proc. of ACM SIGGRAPH'00* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 183–192.

[BP04] Bevacqua E., Pelachaud C.: Expressive audio-visual speech. *Journal of Visualization and Computer Animation 15*, 3-4 (2004), 297–304.

[Bra99] Brand M.: Voice puppery. *Proceedings of ACM SIGGRAPH'99* (1999), pp. 21–28.

[BV99] Blanz V., Vetter T.: A morphable model for the synthesis of 3D faces. *Proceedings of ACM SIGGRAPH'99* (1999), pp. 187–194.

[CB05] Chuang E., Bregler C.: Moodswings: Expressive speech animation. *ACM Transactions on Graphics 24*, 2 (2005), 331–347.

[CDB02] Chuang E. S., Deshpande H., Bregler C.: Facial expression space learning. In *Proceedings of Pacific Graphics'2002* (2002), pp. 68–76.

[CFKP04] Cao Y., Faloutsos P., Kohler E., Pighin F.: Real-time speech motion synthesis from recorded motions. In *Proceedings of Symposium on Computer Animation* (2004), pp. 345–353.

[CFP03] Cao Y., Faloutsos P., Pighin F.: Unsupervised learning for speech motion editing. In *Proceedings of Symposium on Computer Animation* (2003), pp. 225–231.

[CG00] Cosatto E., Graf H. P.: Audio-visual unit selection for the synthesis of photo-realistic talking-heads. In *Proceedings of ICME* (2000), pp. 619–622.

[CM93] Cohen M. M., Massaro D. W.: Modeling coarticulation in synthetic visual speech. In *Models and Techniques in Computer Animation*, *Springer Verlag*, N. M. Thalmann, D. Thalmann (eds.), (1993), 139–156.

[cmu05] http://www.speech.cs.cmu.edu/cgi-bin/cmudict, 2005.

[DCFN06] DENG Z., CHIANG P. Y., FOX P., NEUMANN U.: Animating blendshape faces by cross-mapping motion capture data. In *Proceedings of ACM SIGGGRAPH Symposium on Interactive 3D Graphics and Games* (2006), pp. 43–48.

[DLN05] DENG Z., LEWIS J. P., NEUMANN U.: Synthesizing speech animation by learning compact speech co-articulation models. In *Proceedings of Computer Graphics International* (2005), pp. 19–25.

[DN06] DENG Z., NEUMANN U.: eFASE: Expressive facial animation synthesis and editing with phoneme-level controls. In *Proceedings of Symposium on Computer Animation* (Vienna, Austria, 2006), Eurographics Association, pp. 251–259.

[DN07a] DENG Z., NEUMANN U.: *Data-Driven 3D Facial Animation*. Springer-Verlag Press, 2007.

[DN07b] DENG Z., NOH J. Y.: Computer facial animation: A survey. In *Data-Driven 3D Facial Animation* (Eds: Z. Deng and U. Neumann) (2007), Springer Press, pp. 1–28.

[DNL*06] DENG Z., NEUMANN U., LEWIS J. P., KIM T. Y., BULUT M., NARAYANAN S.: Expressive facial animation synthesis by learning speech co-articulations and expression spaces. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (2006), 1523–1534.

[EGP02] EZZAT T., GEIGER G., POGGIO T.: Trainable videorealistic speech animation. *ACM Transactions on Graphics 21*, 3 (2002), 388–398.

[fes04] http://www.cstr.ed.ac.uk/projects/festival/, 2004.

[GGW*98] GUENTER B., GRIMM C., WOOD D., MALVAR H., PIGHIN F.: Making faces. *Proceedings of ACM SIGGRAPH'98* (1998), pp. 55–66.

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIĆ Z.: Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3 (2004), 522–531.

[HRF01] HASTIE T., RIBSHIRANI R., FRIEDMAN J.: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[KG04] KOVAR L., GLEICHER M.: Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics 23*, 3 (2004), 559–568.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Transactions on Graphics 21*, 3 (2002), 473–482.

[KGT01] KSHIRSAGAR S., GARCHERY S., THALMANN N. M.: Feature point based mesh deformation applied to mpeg-4 facial animation. In *DEFORM '00/AVATARS'00: Proceedings of AVATARS'2000 Workshop on Deformable Avatars* (Deventer, The Netherlands, The Netherlands, 2001), Kluwer, B.V., pp. 24–34.

[KHS01] KÄHLER K., HABER J., SEIDEL H. P.: Geometry-based muscle modeling for facial animation. In *Proceedings of Graphics Interface'2001* (Toronto, Ont., Canada, Canada, 2001), Canadian Information Processing Society, pp. 37–46.

[KP05] KING S. A., PARENT R. E.: Creating speech-synchronized animation. *IEEE Transactions on Visual Graphics 11*, 3 (2005), 341–352.

[KT03] KSHIRSAGAR S., THALMANN N. M.: Visyllable based speech animation. *Computer Graphics Forum 22*, 3 (2003), 631–639.

[LCF00] LEWIS J. P., CORDNER M., FONG N.: Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of ACM SIGGRAPH'2000* (2000), pp. 165–172.

[Lew91] LEWIS J. P.: Automated lip-sync: Background and techniques. *Journal of Visualization and Computer Animation 2*, 4 (1991), 118–122.

[LTW95] LEE Y. C., TERZOPOULOS D., WATERS K.: Realistic modeling for facial animation. *Proceedings of ACM SIGGRAPH'95* (1995), pp. 55–62.

[LWS02] LI Y., WANG T., SHUM H. Y.: Motion texture: a two-level statistical model for character motion synthesis. *ACM Transactions on Graphics 21*, 3 (2002), 465–472.

[MCP*04] MA J., COLE R., PELLOM B., WARD W., WISE B.: Accurate automatic visible speech synthesis of arbitrary 3D model based on concatenation of diviseme motion capture data. *Computer Animation and Virtual Worlds 15* (2004), 1–17.

[MCP*05] MA J., COLE R., PELLOM B., WARD W., WISE B.: Accurate visible speech synthesis based on concatenating variable length motion capture data. *IEEE Transactions on Visualization and Computer Graphics 15*, 5 (2005), 485–500.

[NN01] NOH J. Y., NEUMANN U.: Expression cloning. In *Proceedings of ACM SIGGRAPH'01* (2001), pp. 277–288.

[PB02] PULLEN K., BREGLER C.: Motion capture assisted animation: Texturing and synthesis. *ACM Transactions on Graphics 21*, 3 (2002), 501–508.

[Pel91] PELACHAUD C.: *Communication and Coarticulation in Facial Animation*. PhD Thesis, University of Pennsylvania (1991).

[PHL*98] Pighin F., Hecker J., Lischinski D., Szeliski R., Salesin D. H.: Synthesizing realistic facial expressions from photographs. In *Proceedings of ACM SIGGRAPH'98* (1998), pp. 75–84.

[Pie86] Pierre D. A.: *Optimization Theory with Applications*. Dover Publications, Inc., 1986.

[PKC*03] Pyun H., Kim Y., Chae W., Kang H. W., Shin S. Y.: An example-based approach for facial expression cloning. In *Proceedings of Symposium on Computer Animation* (2003), pp. 167–176.

[PW96] Parke F. I., Waters K.: *Computer Facial Animation*. A K Peters, Wellesley, Massachusets, 1996.

[RCB98] Rose C., Cohen M. F., Bodenheimer B.: Verbs and adverbs: Multidimensional motion interpolation. *IEEE CG&A 18*, 5 (1998), 32–40.

[SHP04] Safonova A., Hodgins J. K., Pollard N. S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions Graphics 23*, 3 (2004), 514–521.

[SNF05] Sifakis E., Neverov I., Fedkiw R.: Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Transactions on Graphics 24*, 3 (2005), 417–425.

[TSL00] Tenenbaum J. B., Silva V. D., Langford J. C.: A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (2000), 2319–2333.

[VBPP05] Vlasic D., Brand M., Pfister H., Popović J.: Face transfer with multilinear models. *ACM Transactions on Graphics 24*, 3 (2005), 426–433.

[WF95] Waters K., Frisble J.: A coordinated muscle model for speech animation. *Proceedings of Graphics Interface'95* (1995), pp. 163–170.

[Wil90] Williams L.: Performance-driven facial animation. In *Proceedings of ACM SIGGRAPH'90* (1990), ACM Press, pp. 235–242.

[WP95] Witkin A., Popović Z.: Motion warping. In *Proceedings of ACM SIGGRAPH '95* (1995), pp. 105–108.

[ZLGS03] Zhang Q., Liu Z., Guo B., Shum H.: Geometry-driven photorealistic facial expression synthesis. In *Proceedings of Symposium on Computer Animation* (2003), pp. 177–186.

[ZSCS04] Zhang L., Snavely N., Curless B., Seitz S. M.: Spacetime faces: High resolution capture for modeling and animation. *ACM Transactions on Graphics 23*, 3 (2004), 548–558.