Animating Lip-Sync Characters with Dominated Animeme Models

Yu-Mei Chen, Fu-Chun Huang, Shuen-Huei Guan, and Bing-Yu Chen, *Member, IEEE*

***Abstract*—Character speech animation is traditionally considered as important but tedious work, especially when taking lip synchronization (lip-sync) into consideration. Although there are some methods proposed to ease the burden on artists to create facial and speech animation, almost none are fast and efficient. In this paper, we introduce a framework for synthesizing lip-sync character speech animation in real time from a given speech sequence and its corresponding texts. Starting from training dominated animeme models for each kind of phoneme by learning the character's animation control signal through an EM-style optimization approach. The dominated animeme models are further decomposed to polynomial-fitted animeme models and corresponding dominance functions while taking *coarticulation* into account. Finally, given a novel speech sequence and its corresponding texts, the animation control signal of the character can be synthesized in real time with the trained dominated animeme models. The synthesized lip-sync animation can even preserve exaggerated characteristics of the character's facial geometry. Moreover, since our method can perform in real time, it can be used for many applications, such as lip-sync animation prototyping, multilingual animation reproduction, avatar speech, mass animation production, etc. Furthermore, the synthesized animation control signal can still be imported into 3D packages for further adjustment, so our method can be easily integrated into existing production pipeline.**

***Index Terms*—lip synchronization, speech animation, character animation, dominated animeme model, animeme modeling, coarticulation.**

I. Introduction

With the popularity of 3D animation and video games, facial and speech character animation is becoming more important than ever. MPEG-4 even defined the facial animation as one of its key features [1]. There are many technologies allowing artists to create high quality character animation, but facial and speech animation is still difficult to sculpt because the correlation and interaction of the muscles on the face are very complicated. Some physically-based simulation methods are provided to approximate the muscles on the face, but the computational cost is very high. A less flexible but affordable alternative is the performance-driven approach [2][3][4][5], in which the motions of an actor is cross-mapped and transferred to a virtual character (see [6] for further discussion). This approach gains much success, but the captured performance is difficult to re-use such that a new performance is required each time when creating a new animation or speech sequence. Manual adjustment is still a popular approach besides the above two, that artists are requested to adjust the face model controls frame-by-frame and compare the results back-and-forth.

When creating facial and speech character animation, it is more challenging to have a character model's lips synchronized. It is a labor-consuming process, and even requires millisecond-precise key-framing. Given a spoken script, the artist has to first match the lips' shapes with their supposed positions. The *transitions* from word-to-word or phoneme-to-phoneme, a.k.a. *coarticulation*, play a major role in speech animation and need to be adjusted carefully [7][**?**]. *Coarticulation* is the phenomenon that a phoneme can influence the mouth shapes of the previous and next phonemes. In other words, the mouth shape depends on not only the current phoneme but also its context, including at least the previous and next phonemes. As opposed to simple articulated animation which can be key-framed with linear techniques, *coarticulation* is non-linear and difficult to model.

In this paper, we propose a framework to synthesize lip-sync character speech animation in real time. For each phoneme, one or multiple **dominated animeme models (DAMs)** are first learned via clustering from a training set of speech-to-animation control signal (e.g. the character controls used in Maya or cross-mapped mocap lip-motions). A **DAM** is the product of a latent dominance function and an intrinsic animime function, where the former controls *coarticulation* and the latter models the mouth shape in the sub-phoneme accuracy. The two entangled functions are learned and decomposed through an EM-style solver.

In the synthesis phase, given a novel speech sequence, the **DAMs** are used to synthesize the corresponding speech-to-animation control signal to generate the lip-sync character speech animation automatically, so it can be integrated into existing animation production pipeline easily. Moreover, since our method can synthesize acceptable and robust lip-sync animation in real time, it can be used in many applications for which prior techniques are too slow, such as lip-sync animation prototyping, multilingual animation reproduction, avatar speech, mass animation production, etc.

To summarize the contributions of this paper:

1) A framework is proposed to synthesize lip-sync character speech animation in real time.
2) Instead of generating hard-to-adjust vertex deformations like other approaches, high-level control signal of 3D character models is synthesized. Hence, our synthesis process can be more easily integrated into existing

Y.-M. Chen and B.-Y. Chen are with National Taiwan University.
F.-C. Huang is with University of California at Berkeley.
S.-H. Guan is with National Taiwan University and Digimax.

Manuscript received August 29, 2011; revised January 30, 2012.

animation production pipeline.

3) We present the DAM, which fits *coarticulation* better by modeling the animation control signal in sub-phoneme precision with the product of a latent dominance function and an intrinsic animeme function.
4) Multiple DAMs are used to handle large intra-animeme variations.

II. Related Work

Face modeling and facial/speech animation are broad topics; [6][7][**?**] provided good surveys. In this section, we separate the face modeling and the specific modeling for lips in the discussion.

A. Facial Animation and Modeling

Most facial animation and modeling methods can be categorized into parameterized/blend-shape, physically-based, data-driven, and machine-learning approaches. For parameterized/blend-shape modeling, faces are parameterized into controls; the synthesis is done manually or automatically via control adjustment. Previous work on linear blend-shape [8][9][10], face capturing/manipulation (FaceIK) [11], and face cloning/cross-mapping [12][13][14][15][16] provided a fundamental guideline for many extensions. However, their underlying mathematical frameworks indeed have some limitations, e.g. the faces outside the span of examples or parameters cannot be realistically synthesized, and these techniques require an excessive number of examples. Other methods reduce the interference between the blend-shapes [17] or enhance the capabilities of cross-mapping to animate the face models [18].

Physically-based methods [19][20] simulate the muscles on the face, and the underlying interaction forms the subtle motion on the skin. The advantage of the physically-based methods over the parameterized/blend-shape ones is extensibility: the faces can be animated more realistically, and the framework allows interaction with objects. However, the muscle-simulation is very expensive, and hence reduces the applicability to interactive controller.

Data-driven methods [21] construct a database from a very large training dataset of faces. The synthesis of novel facial animation is generated by searching the database and minimizing the discontinuity between successive frames. Given the starting and ending example frames, the connecting path in the database forms newly synthesized facial animation. However, they have to deal with missing training data or repetitive occurrence of the same records.

Machine-learning techniques base their capabilities on the learned statistical parameters from the training samples. Previous methods [22][23][24][25] employed various mathematical models and can generate new faces from the learned statistics while respecting the given sparse observations of the new data.

In our system, we adopt the blend-shape facial basis based on “Facial Action Coding System (FACS) [8]” to form the speech-to-animation controls to drive the 3D face models easily. By merging the advantages of the data-driven and machine-learning techniques, we construct a lip-shape motion control database to drive speech activities and moreover generate new lip-sync motions. Unlike other previous methods which directly use training data to synthesize results, our approach can synthesize natural lip shapes that did not appear in the training data set.

B. Lip-Sync Speech Animation

Many speech animation methods derive from the facial animation and modeling techniques. The analysis of the **phonemes** under the context of speech-to-face correspondence, a.k.a. **viseme**, is the subject of much successful work. Many previous methods addressed this issue with Spline generation, path-finding, or signal concatenation.

Parameterized/blend-shape techniques [26][27][28] for speech animation are the most popular methods because of their simplicity. Sifakis *et al.* [29] presented a physically-based approach to simulate the speech controls based on [20] for muscle activation. This method can interact with objects while simulating, but still, the problem is the simulation cost. Data-driven approaches [21][30] form a graph for searching the given sentences. Like similar approaches, they used various techniques, e.g. dynamic programming, to optimize the searching process. Nevertheless, they still suffer from missing data or duplicate occurrence. Machine-learning methods [31][32][33][34][35] learn the statistics for phoneme-to-animation correspondences, which is called **animeme**, in order to connect animation up to speech directly and reduce these searching efforts.

Löfqvist [36] and Cohen and Massaro [37] provided a key insight to decompose the speech animation signal into target values (mouth shapes) and latent dominance functions to model the implicit *coarticulation*. In subsequent work, the dominance functions are sometimes reduced to a diphone or triphone model [33] for simplicity. However, the original framework shows some examples (e.g. the time-locked or look-ahead model) that are difficult to explain by the simpler diphone or triphone model. Their methods are later extended by Cosi *et al.* [38] with the resistance functions and shape functions, which is the basic concept of the **animeme**.

Some recent methods [29][34][35] used the concept of **animeme**, a shape function, to model the sub-viseme signal to increase the accuracy of phoneme fitting. Kim and Ko [34] extended [31] by modeling the **viseme** within a smaller sub-phoneme range with a data-driven approach. *Coarticulation* is modeled via a smooth function in their regularization with the parameters found empirically. However, it has to resolve conflicting and insufficient records in the training set. Sifakis *et al.* [29] modeled the muscle-control-signal animeme (they call it **physeme**) for each phoneme, and concatenate these animemes for words. They found that each phoneme has various similar animemes with slight variations due to *coarticulation*, which is modeled with linear cross-fade weighting in a diphone or triphone fashion.

Kshirsagar *et al.* [39] presented a different approach to model *coarticulation* by using the **visyllable**. Each syllable contains at least one vowel and one or more consonants. It requires about 900 demi-visyllables for the system in their experiments, and therefore the approach needs a huge database.

Wampler *et al.* [35] extended the multilinear face model [25] to derive new lip-shapes for a single face model. *Coarticulation* is modeled by minimizing the lips' positions and forces exerted. However, it is usually unnecessary to sample the face tensor space to produce a single speech segment. Moreover, the face tensor space also inherits the curse of dimensionality, which is also a difficult topic for facial capturing.

We learned from many successful previous methods and improved the deficiencies in them. The analysis in the **sub-viseme**, or so-called **animeme**, space has significant improvements over the viseme analysis. In addition, we also solve for the hidden dominance functions, and extend *coarticulation* beyond the simpler diphone or triphone model. Moreover, the synthesis process is much simpler and faster because the models used for generating the results are trained in an offline pre-pass.

III. Dominated Animeme Models (DAMs)

To animate a character (face) model from a given script (phonemes), it is necessary to form the relationship between the phonemes and animation control signal $C(t)$, which is called **animeme** that means the animation representation of the phoneme. However, due to *coarticulation*, it is hard to model the animeme with a simple function, so we model the animation control signal $C(t)$ with a product of two functions: the animeme function and its dominance function. The animeme function controls the intrinsic mouth shapes when used alone without other phonemes. When putting words together, it is necessary to concatenate several phonemes together, and the dominance functions of the animemes control their individual influence and fall-off, and hence *coarticulation*. Mathematically, one **dominated animeme model (DAM)** is modeled as:

$$C(t) = D(t)A(t), \qquad t \in [-\infty, \infty]$$

where the animeme function $A(t)$ is modeled with a high degree polynomial function to simulate the relationship between phonemes and lip shapes, and the dominance function $D(t)$ is modeled via a modified exponential function, which is used to simulate *coarticulation*.

Some previous literatures [33][36] described the dominance function as a bell-shape function. That means, although our lip-shape is mainly affected by the current phoneme, the lip-shape is also affected by the neighboring phonemes. Inspired by [37], if the time is within the activation of the phoneme (i.e., $t \in [0, 1]$), then the animeme has full influence. Exponential fall-off is applied when time is outside the activation period of the phoneme:

$$D(t) = \begin{cases} 1, & t \in [0,1] \\ \exp\left(\dfrac{-t^2}{\sigma^2 + \varepsilon}\right), & t < 0 \\ \exp\left(\dfrac{-(t-1)^2}{\sigma^2 + \varepsilon}\right), & t > 1 \end{cases} \tag{1}$$

where σ is the phoneme specific parameter affecting the range of influence, and ε is a small constant to prevent dividing by zero.

Putting multiple phonemes together to get the full sequence of animation control signal, we simply concatenate these DAMs with the summation of their normalized values:

$$C^*(t) = \sum_{j=1}^{J} C_j(t_j) = \sum_j D_j(t_j)A_j(t_j), \tag{2}$$

where $j = 1, 2, ..., J$ indicates the j-th phoneme in the given phoneme sequence, and $t_j = (t - s_j)/d_j$ is the normalized local time for each phoneme activation, where s_j is the starting time-stamp of the j-th phoneme and d_j is its duration. Generally, in the dominance function of an animeme, the fall-off controls its influence beyond its phoneme activation period. Strong *coarticulation* has slow fall-off and vice versa. Note that the phonemes farther away from the current phoneme may have very little contribution to it, so the influence of the DAMs far from it is relatively small.

One major observation besides the above description is the intra-animeme variations. In fact, some phonemes strongly depend on lip-shapes than others. By performing the unsupervised clustering [40], we found some phonemes can have multiple DAMs which we call them *modes*; the choice of which *mode* to use depends on the speech context. This finding coincidentally agrees with many successful data-driven methods.

In the subsequent sections, we will use the DAMs and give a system that learns and synthesizes speech animation sequences. To learn the parameters for modeling animemes and their dominance functions, multiple *modes* of each phoneme are first found by affinity-propagation [40]. Then, an EM-style solver is performed to learn the DAM parameters for each *mode*, specifically the polynomial coefficients for animeme functions and the fall-off controls (σ in Eq. (1)) for dominance functions. Once the parameters are learned, we can synthesize the animation control signal given a novel speech sequence and its corresponding texts. The given texts provide the guide to choose an individual DAM for each phoneme, and the chosen DAMs are then concatenated with Eq. (2).

IV. Overview

Fig. 1 shows our system flowchart. The system has two phases: training (left) and synthesis (right). In the training phase, the system takes as input the captured lip-motions or the animation control signal of the character (face) model directly. The animation control signal is usually used to drive the motions of a character model in the modeling tools, like Maya. If we choose the lip-tracking result from a speech video or 3D lip-motions captured by a mocap facility, the data in the vertex domain will be first cross-mapped to the control signal domain (discussed in Appendix A). If there exists acceptable lip-sync character animation, the capturing and cross-mapping processes can be omitted; the speech-to-animation control signal from the existed artist-sculpted or captured speech animation can be used directly.

Then, the speech and its corresponding texts are aligned with SPHINX-II [41] to obtain the aligned scripts (phoneme sequence), which contain phonemes with their starting time-stamps and durations in the speech. The aligned scripts and

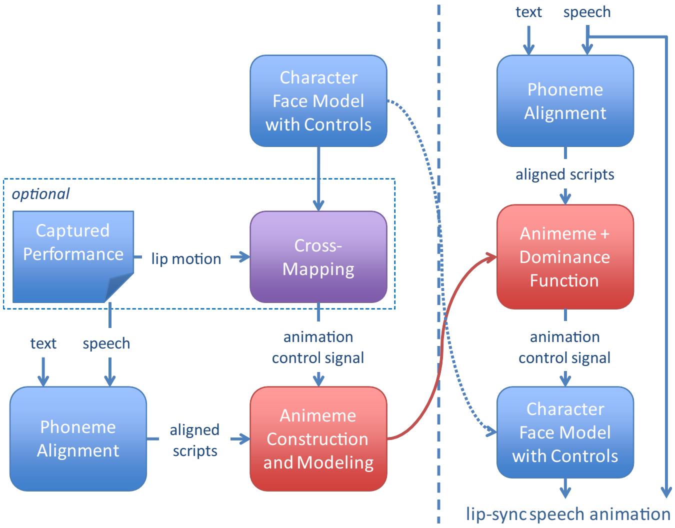

Fig. 1. System flowchart.

animation control signal $C(t)$ are used as training examples to construct the DAMs (Section V) for future novel speech animation synthesis.

In the synthesis phase, we take as input a novel speech and its corresponding texts, and use SPHINX-II again to obtain the aligned scripts. From the scripts, the DAMs are concatenated to generate the animation control signal C^* (Section VI). Finally, the animation control signal C^* is used to animate the character (face) model in Maya or similar modeling tools to generate the lip-sync character speech animation.

The core components in the system are the learning module for constructing and modeling the DAMs and the synthesis module for generating the animation control signal C^*, which will be explained in the next two sections, respectively.

V. Learning DAMs

A. Learning Modes for Phonemes

According to the aligned scripts (phoneme sequence), every phoneme can have many corresponding animation control signals. Based on these training examples, we can construct the phoneme's DAM(s). However, we found it is difficult to decouple the animeme function and its dominance function gracefully if we construct a single DAM for each phoneme due to large intra-animeme variations. Instead, multiple DAMs, or *modes*, for each phoneme are used. The choice of *modes* in a speech sequence depends on the speech context.

The training animation control signal for each phoneme is first fitted and reconstructed with a cubic Spline interpolation, while the duration of the phoneme is parameterized to $t \in [0, 1]$. Then, an unsupervised clustering algorithm, affinity propagation [40], is used to cluster the training control signal into some *modes*; the quantity of the clustering is determined automatically.

Note that the idea of *modes* is not new; data-driven approaches synthesize animation by searching animation clips within the database. This kind of methods has to deal with repetitive clips. The use of which clips depends on smooth transitions and user-specified constraints, which are similar to our choices of *modes*. In the synthesis phase (Section VI), we will discuss the *mode*-selection in more details.

B. Estimating Animeme Function

Assuming each *mode* of each phoneme appears in the sequence exactly only once and denoting the j-th dominance function $D_j(i)$ at time i as a fixed value D_j^i, the estimation of the polynomial function $A_j(t)$ can be reduced to find the polynomial coefficients $a_j^0, a_j^1, ..., a_j^M$. Then, Eq. (2) can be rewritten as:

$$C(i) = \sum_{j=1}^{J} D_j^i \left[\sum_{m=0}^{M} a_j^m (t_j^i)^m \right], \qquad (3)$$

where $t_j^i = (i - s_j)/d_j$ is the normalized local time-stamp from the activation of the j-th phoneme.

Since we want to find the coefficients $a_j^0, a_j^1, ..., a_j^M$ for each phoneme j ($M = 4$ in our implementation), in a regression manner, we can set the partial derivative of regression error $\mathbf{R}$ with respect to the m-th coefficient a_j^m for the j-th phoneme to zero. The least square fitting for regression is:

$$\begin{aligned} f_i &= C(i) - \sum_{j=1}^{J} D_j^i \left[\sum_{m=0}^{M} a_j^m (t_j^i)^m \right], \\ \mathbf{R} = \mathbf{F}^T\mathbf{F} &= \sum_{i=0}^{n} \left(C(i) - \sum_{j=1}^{J} D_j^i \left[\sum_{m=0}^{M} a_j^m (t_j^i)^m \right] \right)^2 \end{aligned} \qquad (4)$$

where $\mathbf{F}$ is the column-concatenated vector formed for each element f_i. Since the unknowns a_j^m are linear in $\mathbf{F}$, the problem is essentially a linear least-square fitting. By setting all partial derivatives to zero and arranging Eq. (4), we can obtain the following matrix representation:

$$\mathbf{D} = \begin{bmatrix} D_1^1(t_1^1)^0 & \cdots & D_1^1(t_1^1)^M & \cdots & D_J^1 & \cdots & D_J^1(t_J^1)^M \\ D_1^2(t_1^2)^0 & \cdots & D_1^2(t_1^2)^M & \cdots & D_J^2 & \cdots & D_J^2(t_J^2)^M \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ D_1^n(t_1^n)^0 & \cdots & D_1^n(t_1^n)^M & \cdots & D_J^n & \cdots & D_J^n(t_J^n)^M \end{bmatrix},$$

$$\mathbf{A} = \begin{bmatrix} a_1^0 & \cdots & a_1^M & \cdots & a_J^0 & \cdots & a_J^M \end{bmatrix}^T,$$

$$\mathbf{C} = \begin{bmatrix} C^0 & C^1 & C^2 & \cdots & C^n \end{bmatrix}^T,$$

where $\mathbf{D}$ is the dominance matrix, $\mathbf{A}$ is the coefficient vector we want to solve, and $\mathbf{C}$ is the observed values at each time i, so the minimum error to the regression fitting can be written in the standard normal equation with the following matrix form:

$$(\mathbf{D}^T\mathbf{D})\mathbf{A} = \mathbf{D}^T\mathbf{C}, \qquad (5)$$

where $\mathbf{D}$ is an $n \times ((M+1) \times J)$ matrix, $\mathbf{C}$ is an n vector, and $\mathbf{A}$ is an $(M+1) \times J$ vector to be solved.

If we remove the assumption that each *mode* of each phoneme appears exactly once, multiple occurrences of each *mode* of a phoneme have to be fitted to the same value. Hence, we can rearrange the multiple occurring terms and make it easier to solve. For example, if phoneme 1 (with only one *mode*) appears twice as the first and third phonemes in the phoneme sequence, then Eq. (3) becomes:

$$\begin{aligned} C(i) &= D^i_{1_1} A_{1_1}(t^i_{1_1}) + D^i_2 A_2(t^i_2) + D^i_{1_2} A_{1_2}(t^i_{1_2}) + ... \\ &= \left[D^i_{1_1} + D^i_{1_2}\right] a^0_1 + \left[D^i_{1_1}(t^i_{1_1}) + D^i_{1_2}(t^i_{1_2})\right] a^1_1 + ... \\ &+ \ D^i_2 a^0_2 + D^i_2 a^1_2(t^i_2) + D^i_2 a^2_2(t^i_2)^2 + ..., \qquad (6) \end{aligned}$$

where 1_1 and 1_2 indicate the first and second times the phoneme 1 appeared. Note that the polynomial coefficients a^m_j of the animeme function $A_j(t)$ are the same and independent to the occurrences.

By the above re-arrangement, we can remove the original assumption that each *mode* of each phoneme can appear exactly only once, and rewrite the original term in Eq. (3) with the summation of each occurrence H_j of the same *mode* of phoneme j as:

$$D^i_j(t^i_j)^m \Rightarrow \sum_{H_j} D^i_{j_h}(t^i_{j_h})^m, \qquad (7)$$

where j_h denotes the h-th time occurrence of the *mode* of phoneme j.

C. Estimating Dominance Function

In the previous section to estimate the animeme function $A_j(t)$ of the j-th phoneme, we assumed that its dominance function $D_j(t)$ is known and fixed. In this section, we will describe how to estimate the dominance function $D_j(t)$ over the regression, given that the animeme value $A_j(i)$ at time i is known and fixed, denoted as A^i_j. Back to the definition of the dominance function formulated in Eq. (1), for phoneme j, its influence control is affected by σ_j, which is unknown now.

Here, we want to minimize the regression Eq. (4) again as we did in the previous section. However, since the parameter σ_j for regression is non-linear, we need a more sophisticated solver. Standard Gauss-Newton iterative solver is used to approach the minimum of the regression error $\mathbf{R}$. As we defined the residual error in the previous section, the Gauss-Newton algorithm linearizes the residual error as:

$$\begin{aligned} f_i &= C(i) - \sum_{j=1}^{J} D_j(t^i_j) A^i_j, \\ \mathbf{F}(\sigma_j + \delta) &\approx \mathbf{F}(\sigma_j) + \mathbf{J}\delta, \qquad (8) \end{aligned}$$

where $t^i_j = (i - s_j)/d_j$ is the normalized local time, $\mathbf{F}$ is formed by f_i but takes as input the influence control σ_j for the j-th phoneme, δ is the updating step for gradient direction of the Gauss-Newton solver, and $\mathbf{J}$ is the Jacobian matrix. Each iteration of the Gauss-Newton algorithm solves a linearized problem to Eq. (4), and after removing the terms that do not dependent on δ, we get the follows:

$$\begin{aligned} \mathbf{J}^T\, \mathbf{J}\, \delta &= -\mathbf{J}^T\, \mathbf{F}, \\ \sigma^{k+1}_j &= \sigma^k_j + \delta. \qquad (9) \end{aligned}$$

The Gauss-Newton algorithm repeatedly optimizes the regression error by updating δ to σ^k_j at the k-th iteration, and achieves linear convergence.

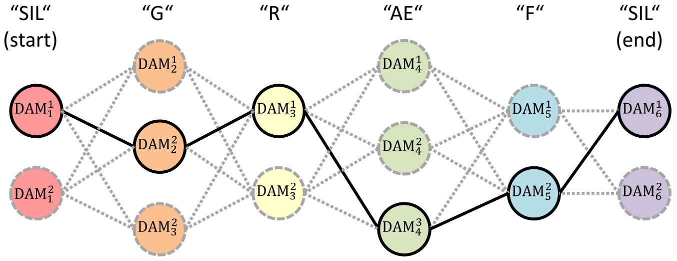

Fig. 2. An animeme-graph example for synthesizing “Graph”. There are multiple DAMs (modes) for one phoneme (with the same color). The suitable sequence (denoted by solid circles and lines) is selected by A* algorithm.

D. Learning with Iterative Optimization

In the previous two sections, we showed how to minimize the regression error by estimating the animeme function $A_j(t)$ and its hidden dominance function $D_j(t)$. Since the entire formulation is not linear and cannot be solved intuitively, we employed an EM-style strategy that iterates between the estimation of the animeme function $A_j(t)$ and the optimization for the dominance function $D_j(t)$.

- The **E-step** involves estimating the polynomial coefficients a^m_j for each animeme function $A_j(t)$ by solving a linear regression using the standard normal equation.
- The **M-step** tries minimizing the regression error to estimate the influence controls σ_j by improving the non-linear dominance function $D_j(t)$.

First, when solving for the **E-step**, the initial influence control parameters σ_j involved in $D_j(t)$ are set to 1. At the **M-step**, where the Gauss-Newton algorithm linearizes the function by iteratively updating the influence controls σ_j, all parameters of the polynomial coefficients a^m_j are carried from the first half of the iteration. The EM-style strategy keeps iterating between the **E-step** and **M-step** until no more improvement on regression error can be done. Convergence of optimizing $D_j(t)$ is fast, but the effect of estimating $A_j(t)$ has more perturbation on σ_j. The number of iterations for convergence is varying for different DAMs, which is directly proportional to the quantity of the clustered control signals for each DAM, but the process is an off-line computation in the training phase separate from synthesis.

VI. Synthesizing with DAMs

In the synthesis phase, we want to generate the output control signal according to the input phoneme sequence. Since some phonemes may have multiple *modes*, we have to decide which *mode* should be used for each phoneme. The goal to construct the output animation control signal requires selecting the most suitable *mode* for each phoneme, and then directly use Eq. (2) to concatenate the DAMs in the sequence.

Giving a phoneme sequence $j = 1, 2, ..., J$ and possible *modes* DAM^g_j ($g = 1, ..., G_j$, where G_j is the number of *modes*) for each phoneme j, the animemes can form an animeme-graph as shown in Fig. 2. The selection of suitable *modes* for the phoneme sequence can be treated as a graph search problem, and A* algorithm is used in our implementation. Since we want to find a compromise between the

TABLE I
THE MODELS USED IN THIS PAPER.

model	vertex#	face#	control#
Afro-woman	5,234	5,075	7
Boy	6,775	6,736	7
Child	6,991	6,954	16
Old-hero	8,883	8,738	8
Court-lady	1,306	1,307	7

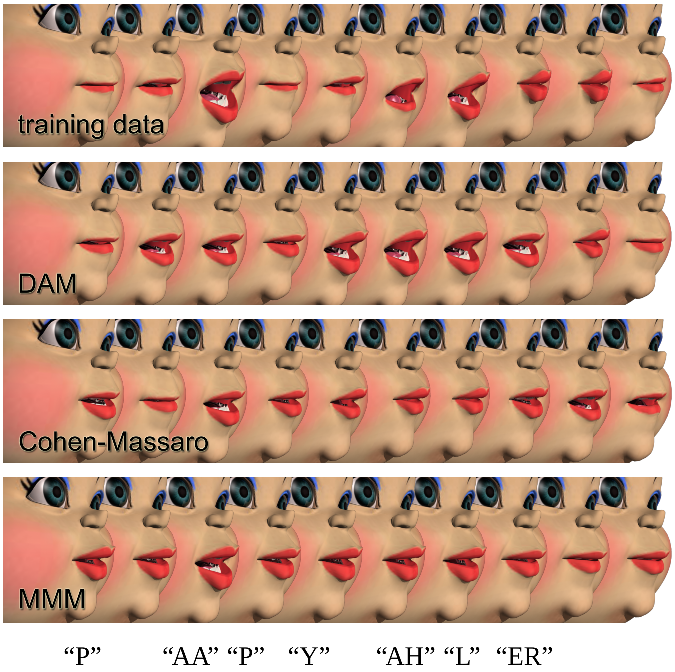

Fig. 3. The comparison of training data (the 1st raw) and the synthesized results of DAM (the 2nd raw), Cohen-Massaro model (the 3rd raw), and MMM (the 4th raw), while speaking "popular" by Afro-woman.

likelihood of the *modes* and the smoothness in the animation, the cost of each node in the animeme-graph is set as:

$$E = w_c E_c + w_s E_s, \qquad (10)$$

where E_c is a data term, which represents the likelihood of the *mode* DAM_j^g in the training set linked with its previous and next phonemes, E_s is the smoothness term computing the C^2 smoothness on the joint frame of every DAM_j^g $(g = 1, ..., G_j)$ of the current phoneme j and every DAM_{j-1}^g $(g = 1, ..., G_{j-1})$ of its previous phoneme $j - 1$, and w_c and w_s are the weights of the error terms. We used $w_c = 1000$ and $w_s = 1$ for all results in this paper.

VII. RESULT

The training set involves 80 sentences and about 10 minutes of speech context with unbiased content. In the training phase, constructing the DAMs costs about 50∼60 minutes per control on a desktop PC with an Intel Core2 Quad Q9400 2.66GHz CPU and 4GB memory. For synthesizing a lip-sync speech animation, the animation control signal formed by our DAMs are generated in real time (i.e., 0.8 ms. per phoneme on average). Table I shows the number of vertices, faces, and controls of each model, respectively, used in this paper.

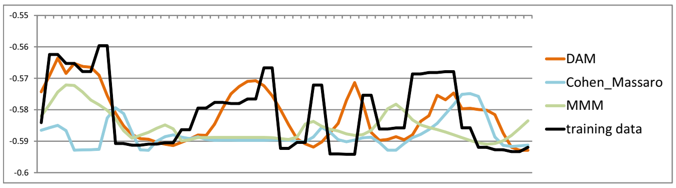

Fig. 4. The comparison of the signal fitted in Fig. 3 by DAM, Cohen-Massaro Model, and MMM with the captured one. The y-axis shows one of the coordinates of a control.

Fig. 3 shows a comparison of the training data and the synthesized results of our **dominated animeme model (DAM)**, Cohen-Massaro model [37], and multi-dimensional morphable model (MMM) [31], while speaking "popular" by using the Afro woman model. Fig. 4 shows a part of signal fitting for these results. The average L^2-norms for DAM, Cohen-Massaro model, and MMM are 0.4724, 0.6297, and 0.5023, respectively. This sequence represents continuous lip motion, and the flow is from left to right. According to the training data, the lips should be closed during the phoneme "P" and opened for other phonemes appropriately. At the last frame of the sequence, the mouth closes to prepare for the next word. Note that the Cohen-Massaro model is implemented using our DAM by setting $M = 0$ in Eq. (4), i.e., the polynomial form is reduced to only the constant term. The formulation of our dominance function (Eq. (1)) is very similar to their original form but with the flexible extension that the shapes of the phonemes can be varied. The reconstruction result of the Cohen-Massaro model is too smooth at some parts in the sequence, such that consecutive phonemes are greatly influenced, i.e., they span too much. Hence, the features of a few phonemes can be shown, but others are not as prominent as they should be. In contrast, our DAM spans more properly in range with respect to the training data. The MMM formulates the fitting and synthesis as a regulation problem. It fits each phoneme as a multidimensional Gaussian distribution and forms the words or sentences as a path going through these phoneme regions by minimizing an energy function containing a target term and a smoothness term. The speech poses using MMM have good timing but lack prominent features.

Fig. 5 shows two results of speaking two words - "apple" and "girl" by using the Afro woman model. As shown in the close-up view of the mouth, although the last phonemes of the two words are the same ("L"), the lip shapes are different due to *coarticulation*. Note that the lip shape for pronouncing the phoneme "P" shows the mouth closes well, although some other similar methods cannot due to the smoothness effect. One can also notice that while pronouncing the phoneme "ER", the tongue is rolled. In general, the shape change of the tongue is very hard to capture. However, since our method uses animation control signal as the training data, once the target model is designed well for performing such features, our synthesis results can also keep these characteristics.

Fig. 6 and 7 show two other results with different models: the old hero speaks "old man" and the boy speaks "top of the world". All of them have their typical styles. Comparing the lip shapes of the two models while pronouncing the

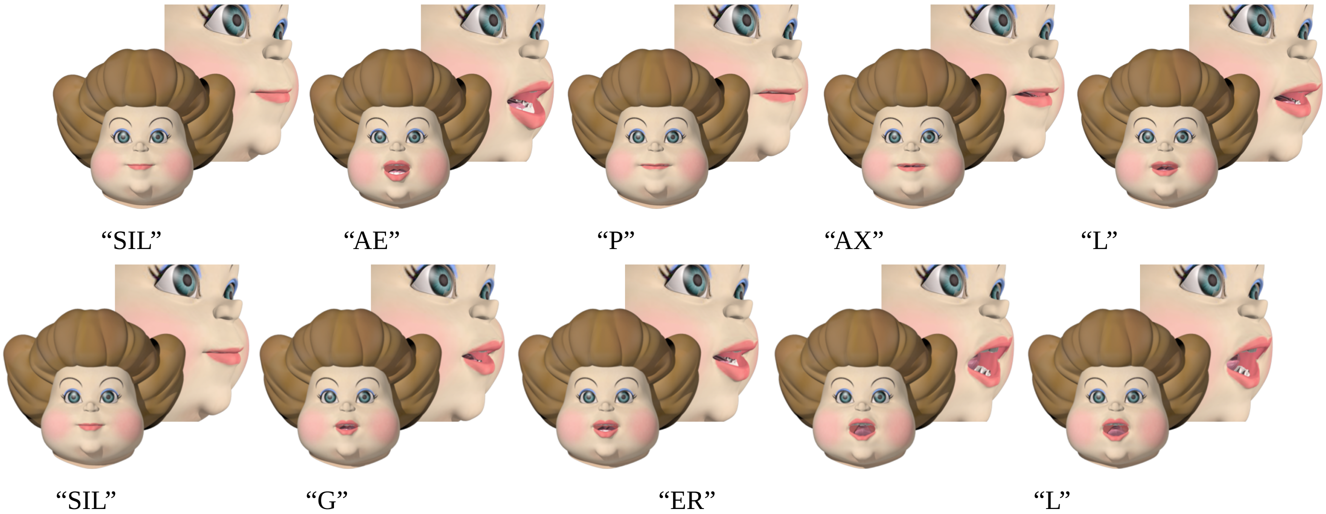

Fig. 5. Result of speaking “Apple” (upper) and “Girl” (lower) by Afro-woman.

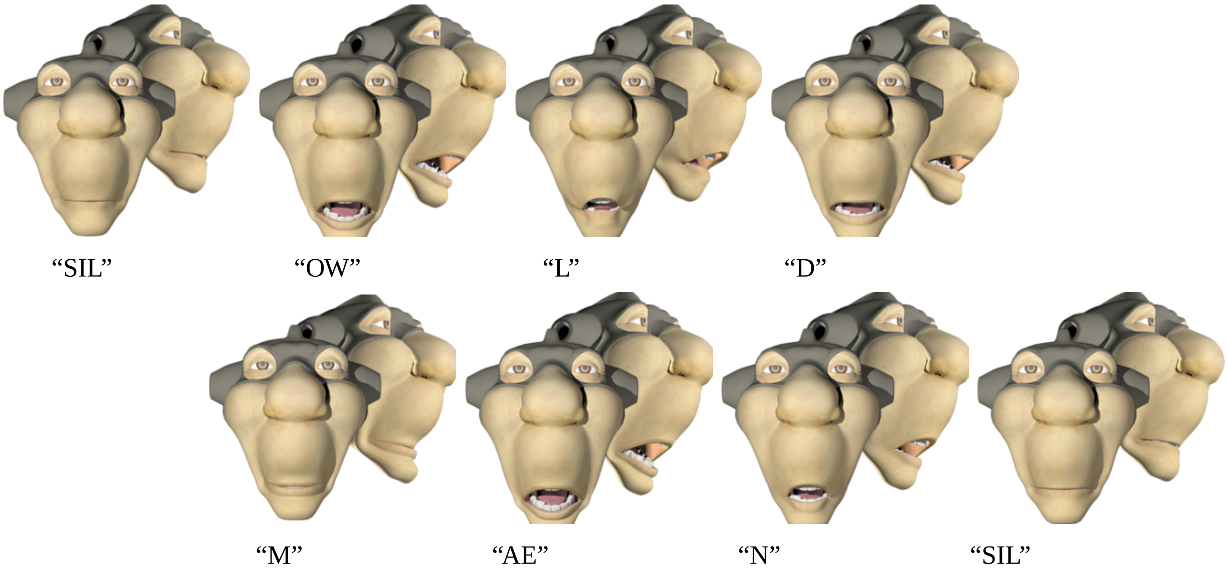

Fig. 6. Result of speaking “Old Man” by Old-hero.

phoneme “L”, their lip shapes performed in two different ways because of *coarticulation* and their characteristics. To keep the models’ characteristics while synthesis, our method is character (model) dependent. Each character’s DAMs should be trained by its own animation control signal. Models with similar controls can use the same DAMs. Of course, artists can also use another model’s trained DAMs to make a prototype of a novel model, and then refine the model for training its own DAMs. This can speed up the training preparation time while keeping the quality of the training data.

VIII. Conclusion and Future Work

In this paper, we propose a new framework for synthesizing lip-sync character speech animation in real time with a given novel speech sequence and its corresponding texts. Our method produces fairly nice transitions in time and generates the animation control parameters that are formed by our **dominated animeme models** (**DAMs**), which are constructed and modeled from the training data in sub-phoneme accuracy for capturing *coarticulation* well. Through an EM-style optimization approach, the DAMs are decomposed to the polynomial-fitted animeme functions and their corresponding dominance functions according to the phonemes. Given a phoneme sequence, the DAMs are used to generate the animation control signal to animate the character (face) model in Maya or similar modeling tools in real time while still keeping the character’s exaggerated characteristics. Moreover, the DAMs are constructed by the character controls instead of absolute lip shapes, so it can perform better training/synthesizing results and is suitable to be integrated into existing animation pipeline. By using the Facial Animation Parameters (FAPs) defined in MPEG-4 for training and synthesis, our approach can be easily extended to support MPEG-4 facial animation [42].

Even though the quality of the synthesized lip-sync char-

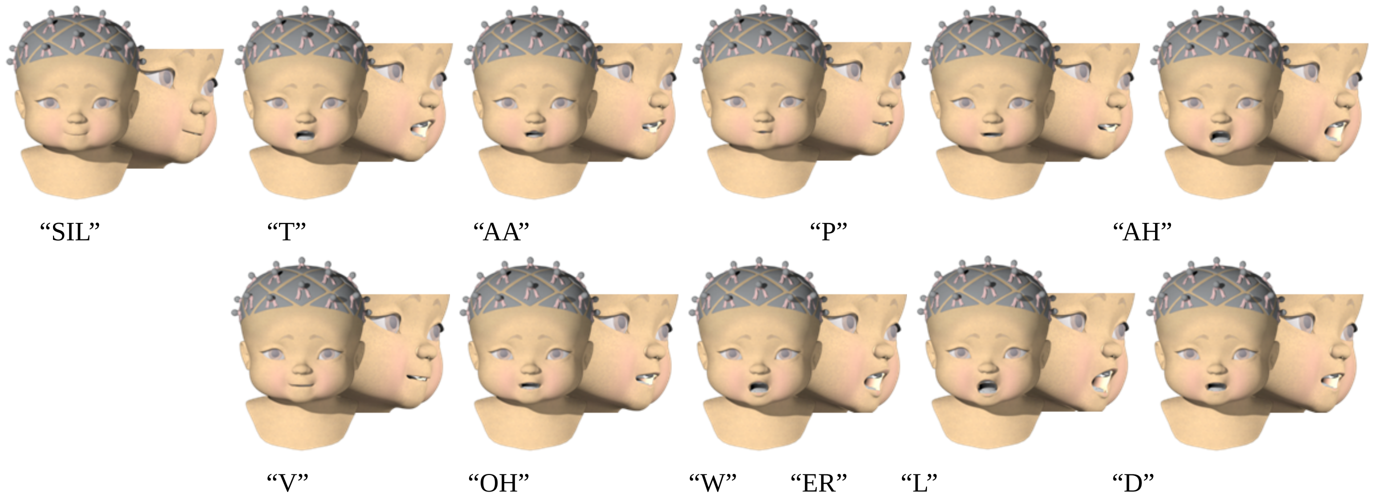

Fig. 7. Result of speaking "Top of The World" by Boy.

acter speech animation may not be perfect as compared with that of animation created manually by an artist, the synthesized animation can still easily be fine-tuned, since the automatically generated animation control signal is lip-synchronized and can be used directly in Maya or similar animation tools. By extending the phoneme dictionary, our method can also be used to produce multilingual lip-sync speech animation easily. Furthermore, since our method can synthesize acceptable and robust lip-sync character animation in real time, it can be used for many applications for which prior methods are inadequate, such as lip-sync animation prototyping, multilingual animation reproduction, avatar speech, mass animation production, etc.

Our model still has some weaknesses, such as that it currently infers the dynamics of motion solely from the training data set. If the training data set does not contain speech similar to the synthesis target, the results may be inaccurate. For example, if the training data set contains only ordinary speech, it will be unsuitable for synthesizing a singing character, because the typical phoneme behavior for singing a song varies greatly from the ordinary speech and imposes more challenges for dynamics modeling. A second weakness is that in our DAMs, we used a function of Gaussian-based form to model the dominance functions. A potential problem is that while sining a song, certain phonemes may extend indefinitely with dragging sounds. It is not only difficult for a speech recognizer to identify the ending time, but also the Gaussian-based form cannot accommodate such effects. One possible solution is to model the dominance functions with greater variability and non-symmetric models.

Appendix A
Cross-Mapping

Although the input of our system is animation control signal, to ease the efforts for adjusting the character (lip) model, we also provide a method to cross-map the captured lip motion to the animation control signal. After the lip motion is captured, the key-lip-shapes L_k are identified first, which can be pointed out by the artist or by using an unsupervised clustering algorithm, e.g. [40]. The captured key-lip-shapes L_k are then used to fit the captured lip motion L^i for each frame i by using the Non-Negative Least Square (NNLS) algorithm to obtain the blending coefficients α_k^i. This process can be expressed as the following constrained minimization:

$$\min \|L^i - \sum_{k=1}^{K} \alpha_k^i L_k\|^2, \quad \forall \alpha_k^i \geq 0,$$

where K is the number of identified key-lip-shapes. The above clustering and fitting process for the captured lip motion needs to be performed only once. If the target character model has some well-defined bases, it is better to assign the key-lip-shapes to the bases manually, since the blending coefficients α_k^i can be used as the control signal C^i directly without further processing.

To cross-map the input captured lip motion to the target character model, the identified key-lip-shapes L_k are first used to guide the artist to adjust the vertices $\mathbf{V}$ on the lips of the target character model to imitate the key-lip-shapes L_k while keeping the character's characteristics. The number of adjusted vertices should be equal to or more than that of the character controls C (i.e., $\|\mathbf{V}\| \geq \|C\|$) for solving the constrained minimization in the next paragraph. Then, the blending coefficients α_k^i are used to blend the adjusted lip vertices $\mathbf{V}_k$ for key-lip-shapes L_k to obtain the lip vertices $\mathbf{V}^i$ for each frame i via:

$$\mathbf{V}^i = \sum_{k=1}^{K} \alpha_k^i \mathbf{V}_k.$$

Instead of using the lip vertices $\mathbf{V}^i$ for training directly, for better training/synthesis results and animation pipeline integration, the training and synthesizing are performing on character controls. Assuming there are $\mathfrak{K}$ animation controls $C_{\mathfrak{k}} \in C$, which can be used to drive the lip motions of the target character model, the animation control signal $C_{\mathfrak{k}}^i$ for each frame i and control $\mathfrak{k}$ can be obtained by fitting the lip vertices $\mathbf{V}^i$ as the following constrained minimization:

$$\min \|\mathbf{V}^i - \sum_{\mathfrak{k}=1}^{\mathfrak{K}} \mathbf{V}(C^i_{\mathfrak{k}})\|^2,$$

where $\mathbf{V}(\cdot)$ denotes the transfer function from control signal to lip vertices, and each animation control $C^i_{\mathfrak{k}} \in C^i$ is constrained to $[0,1]$. Again, it is solved by the NNLS algorithm.

References

[1] G. Abrantes and F. Pereira, "MPEG-4 facial animation technology: survey, implementation, and results," *IEEE TCSVT*, vol. 9, no. 2, pp. 290–305, 1999.

[2] L. Williams, "Performance-driven facial animation," in *Proc. SIGGRAPH'90*, pp. 235–242.

[3] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin, "Making faces," in *Proc. SIGGRAPH'98*, pp. 55–66.

[4] W.-C. Ma, A. Jones, J.-Y. Chiang, T. Hawkins, S. Frederiksen, P. Peers, M. Vukovic, M. Ouhyoung, and P. Debevec, "Facial performance synthesis using deformation-driven polynomial displacement maps," *ACM TOG*, vol. 27, no. 5, p. Article No.: 121, 2008.

[5] T. Weise, S. Bouaziz, H. Li, and M. Pauly, "Realtime performance-based facial animation," *ACM TOG*, vol. 30, no. 4, pp. 77:1–77:10, 2011.

[6] F. Pighin and J. P. Lewis, "Performance-driven facial animation: Introduction," in *SIGGRAPH'06 Course Notes*.

[7] F. I. Parke and K. Waters, *Computer Facial Animation*, 2nd ed. AK Peters, 2008.

[8] P. Ekman and W. V. Friesen, *Manual for the Facial Action Coding System*. Consulting Psychologist Press, 1977.

[9] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin, "Synthesizing realistic facial expressions from photographs," in *Proc. SIGGRAPH'98*, pp. 75–84.

[10] I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. H. Salesin, J. Seims, R. Szeliski, and K. Toyama, "Performance-driven hand-drawn animation," in *Proc. NPAR'00*, pp. 101–108.

[11] L. Zhang, N. Snavely, B. Curless, and S. M. Seitz, "Spacetime faces: high resolution capture for modeling and animation," *ACM TOG*, vol. 23, no. 3, pp. 548–558, 2004.

[12] J.-Y. Noh and U. Neumann, "Expression cloning," in *Proc. SIGGRAPH'01*, pp. 277–288.

[13] H. Pyun, Y. Kim, W. Chae, H. W. Kang, and S. Y. Shin, "An example-based approach for facial expression cloning," in *Proc. SCA'03*, pp. 167–176.

[14] J. Chai, J. Xiao, and J. Hodgins, "Vision-based control of 3d facial animation," in *Proc. SCA'03*, pp. 193–206.

[15] K. Na and M. Jung, "Hierarchical retargetting of fine facial motions," *EG CGF*, vol. 23, no. 3, pp. 687–695, 2004.

[16] R. W. Sumner and J. Popović, "Deformation transfer for triangle meshes," *ACM TOG*, vol. 23, no. 3, pp. 399–405, 2004.

[17] J. P. Lewis, J. Mooser, Z. Deng, and U. Neumann, "Reducing blendshape interference by selected motion attenuation," in *Proc. I3D'05*, pp. 25–29.

[18] Z. Deng, P.-Y. Chiang, P. Fox, and U. Neumann, "Animating blendshape faces by cross-mapping motion capture data," in *Proc. I3D'06*, pp. 43–48.

[19] B. Choe, H. Lee, and H.-S. Ko, "Performance-driven muscle-based facial animation," *JVCA*, vol. 12, no. 2, pp. 67–79, 2001.

[20] E. Sifakis, I. Neverov, and R. Fedkiw, "Automatic determination of facial muscle activations from sparse motion capture marker data," *ACM TOG*, vol. 24, no. 3, pp. 417–425, 2005.

[21] Z. Deng and U. Neumann, "eFASE: Expressive facial animation synthesis and editing with phoneme-isomap control," in *Proc. SCA'06*, pp. 251–259.

[22] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Proc. SIGGRAPH'99*, pp. 187–194.

[23] E. S. Chuang, H. Deshpande, and C. Bregler, "Facial expression space learning," in *Proc. PG'02*, pp. 68–76.

[24] Y. Wang, X. Huang, C.-S. Lee, S. Zhang, Z. Li, D. Samaras, D. Metaxas, A. Elgammal, and P. Huang, "High resolution acquisition, learning and transfer of dynamic 3-d facial expressions," *EG CGF*, vol. 23, no. 3, pp. 677–686, 2004.

[25] D. Vlasic, M. Brand, H. Pfister, and J. Popović, "Face transfer with multilinear models," *ACM TOG*, vol. 24, no. 3, pp. 426–433, 2005.

[26] C. Bregler, M. Covell, and M. Slaney, "Video rewrite: driving visual speech with audio," in *Proc. SIGGRAPH'97*, pp. 353–360.

[27] M. Brand, "Voice puppetry," in *Proc. SIGGRAPH'99*, pp. 21–28.

[28] E. Chuang and C. Bregler, "Mood Swings: expressive speech animation," *ACM TOG*, vol. 24, no. 2, pp. 331–347, 2005.

[29] E. Sifakis, A. Selle, A. Robinson-Mosher, and R. Fedkiw, "Simulating speech with a physics-based facial muscle model," in *Proc. SCA'06*, pp. 261–270.

[30] Y. Cao, P. Faloutsos, E. Kohler, and F. Pighin, "Real-time speech motion synthesis from recorded motions," in *Proc. SCA'04*, pp. 345–353.

[31] T. Ezzat, G. Geiger, and T. Poggio, "Trainable videorealistic speech animation," *ACM TOG*, vol. 21, no. 3, pp. 388–398, 2002.

[32] Y.-J. Chang and T. Ezzat, "Transferable videorealistic speech animation," in *Proc. SCA'05*, pp. 143–151.

[33] Z. Deng, U. Neumann, J. P. Lewis, T.-Y. Kim, M. Bulut, and S. Narayanan, "Expressive facial animation synthesis by learning speech coarticulation and expression spaces," *IEEE TVCG*, vol. 12, no. 6, pp. 1523–1534, 2006.

[34] I.-J. Kim and H.-S. Ko, "3D lip-synch generation with data-faithful machine learning," *EG CGF*, vol. 26, no. 3, pp. 295–301, 2007.

[35] K. Wampler, D. Sasaki, L. Zhang, and Z. Popović, "Dynamic, expressive speech animation from a single mesh," in *Proc. SCA'07*, pp. 53–62.

[36] A. Löfqvist, "Speech as audible gestures," in *Speech Production and Speech Modeling*. Springer, 1990, pp. 289–322.

[37] M. M. Cohen and D. W. Massaro, "Modeling coarticulation in synthetic visual speech," in *Proc. CA'93*, pp. 139–156.

[38] P. Cosi, E. M. Caldognetto, G. Perin, and C. Zmarich, "Labial coarticulation modeling for realistic facial animation," in *Proc. ICMI'02*, pp. 505–510.

[39] S. Kshirsagar and N. Magnenat-Thalmann, "Visyllable based speech animation," *EG CGF*, vol. 22, no. 3, 2003.

[40] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[41] X. Huang, F. Alleva, H.-W. Hon, M.-Y. Hwang, K.-F. Lee, and R. Rosenfeld, "The SPHINX-II speech recognition system: An overview," *CSL*, vol. 7, no. 2, pp. 137–148, 1993.

[42] Y. Zhang, Q. Ji, Z. Zhu, and B. Yi, "Dynamic facial expression analysis and synthesis with MPEG-4 facial animation parameters," *IEEE TCSVT*, vol. 18, no. 10, pp. 1383–1396, 2008.

Yu-Mei Chen received the BS and MS degrees in Information Management and Computer Science and Information Engineering from the National Taiwan University, Taipei, in 2008 and 2010, respectively. Her research interests include mainly for computer animation.

Fu-Chun Huang received the BS and MS degrees in Information Management from the National Taiwan University, Taipei, in 2005 and 2007, respectively. Currently, he is working toward the PhD degree in the Dept. of EECS of the University of California at Berkeley. His research interests include mainly for computer graphics.

Shuen-Huei Guan received the BS and MS degrees in Computer Science and Information Engineering from the National Taiwan University, Taipei, in 2002 and 2004, respectively. Currently, he is working as the R&D manager in Digimax Inc., and pursuing his PhD in the Graduate Inst. of Networking and Multimedia of the National Taiwan University. His research interests include mainly for computer graphics, stereoscopic vision, and computer animation.

Bing-Yu Chen received the BS and MS degrees in Computer Science and Information Engineering from the National Taiwan University, Taipei, in 1995 and 1997, respectively, and received the PhD in Information Science from The University of Tokyo, Japan, in 2003. Currently, he is working as a professor in the National Taiwan University. His research interests include mainly for computer graphics, image and video processing, and human-computer interaction. He is a member of the IEEE, ACM, and Eurographics, and is the Secretary of the Taipei ACM SIGGRAPH since 2010.