A Practical and Configurable Lip Sync Method for Games

Yuyu Xu* Andrew W. Feng† Stacy Marsella‡ Ari Shapiro§

USC Institute for Creative Technologies

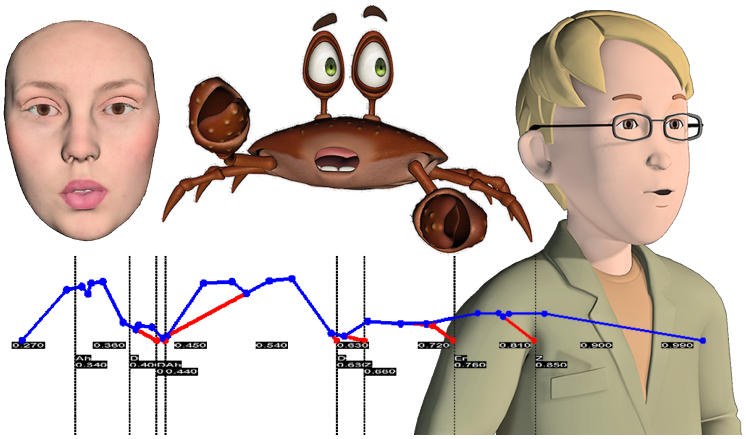

Figure 1: Accurate lip synchronization results for multiple characters can be generated using the same set of phone bigram blend curves. Our method uses animator-driven data to produce high quality lip synchronization in multiple languages. Our method is well-suited for animation pipelines, since it uses static facial poses or blendshapes and can be directly edited by an animator, and modified as needed on a per-character basis.

Abstract

We demonstrate a lip animation (lip sync) algorithm for real-time applications that can be used to generate synchronized facial movements with audio generated from natural speech or a text-to-speech engine. Our method requires an animator to construct animations using a canonical set of visemes for all pairwise combinations of a reduced phoneme set (phone bigrams). These animations are then stitched together to construct the final animation, adding velocity and lip-pose constraints. This method can be applied to any character that uses the same, small set of visemes. Our method can operate efficiently in multiple languages by reusing phone bigram animations that are shared among languages, and specific word sounds can be identified and changed on a per-character basis. Our method uses no machine learning, which offers two advantages over techniques that do: 1) data can be generated for non-human characters whose faces can not be easily retargeted from a human speaker's face, and 2) the specific facial poses or shapes used for animation can be specified during the setup and rigging stage, and before the lip animation stage, thus making it suitable for game pipelines or circumstances where the speech targets poses are predetermined, such as after acquisition from an online 3D marketplace.

Index Terms: I.3.2 [Computing Methodologies]: Computer Graphics—Methodology and Techniques; I.3.7 [Computing Methodologies]: Computer Graphics—Three-Dimensional Graphics and Realism; I.3.8 [Computing Methodologies]: Computer Graphics—Applications

*e-mail: yxu@ict.usc.edu
†e-mail: feng@ict.usc.edu
‡e-mail: marsella@ict.usc.edu
§e-mail: shapiro@ict.usc.edu

1 Motivation

Synchronizing the lip and mouth movements naturally along with animation is an important part of convincing 3D character performance. In this paper, we present a simple, portable and editable lip-synchronization method that works for multiple languages, requires no machine learning, can be constructed by a skilled animator, is effective for real-time simulations such as games, and can be personalized for each character. Our method associates animation curves designed by an animator on a fixed set of static facial poses, with sequential pairs of phonemes (phone bigrams), and then stitches these animations together to create a set of curves for the facial poses along with constraints that ensure that key poses are properly played. Diphone- and triphone-based methods have been explored in various previous works, often requiring machine learning. However, our experiments have shown that animating phoneme pairs (such as phone bigrams or diphones), as opposed to phoneme triples or longer sequences of phonemes, is sufficient for many types of animated characters. Also, our experiments have shown that skilled animators can sufficiently generate the data needed for good quality results. Thus our algorithm does not need any specific rules about coarticulation, such as dominance functions or language rules. Such rules are implicit within the artist-produced data. In order to produce a tractible set of data, our method reduces the full set of 40 English phonemes to a smaller set of 21, which are then annotated by an animator. Once the full animation set has been generated, it can be reused for multiple characters. Each additional character requires a small set of static poses or blendshapes that match the original pose set. Lip sync data needed for a new language requires a new set of phone bigrams for each language,

although similar phonemes among languages can share the same phone bigram curves. We show how to reuse our English phone bigram animation set to adapt to a Mandarin set. Our method works with both natural speech and text-to-speech engines.

Our artist driven approach is useful for lip syncing non-human characters whose lip or face configuration doesn't match a human's, which would make it difficult to retarget. Animators can generate a lip animation data set for any character or class of characters, regardless of the facial setup specified. We demonstrate such an example on an animated, talking crab that we acquired from an online 3D content source, and whose facial configuration does not match any existing set of lip sync poses. In addition, our method can be used with any set of facial poses or blend shapes as input, thus making it suitable for game pipelines where the characters must adhere to specific facial configurations. This differs from many machine learning algorithms where the retargeting efforts must be done once the learned data is generated, or where specific facial poses are output from the machine learning process, but differ according to the input data. By allowing the setup of the character to be determined in advance of generating the lip animation data, our system can be adjusted to be compatible with any existing lip animation setup, thus making it ideally suitable for game pipelines. We demonstrate this capability by using the same facial poses as a commercial lip animation tool, FaceFX [19], whose results we can then directly compare against. In addition, control over our method can be achieved by reanimating specific phoneme pairs (such as L-Oh, as in the word 'low'). Thus control over various parts of the algorithm are both intuitive and controllable.

It is difficult to compare the quality of the results between one lip syncing method to another, since the models, data set, and characteristics are often different between methods. With some exceptions [27], few research methods attempt to compare their own results to other established research methods or commercial results. There are many reasons for this, including the difficulty in reproducing exactly other methods and the lack of availability of data sets. Consequently, comparisons are made to simpler methods [34]. By contrast, we present a direct comparison with a popular commercial lip syncing engine, FaceFX [19], using identical models and face shapes. We thus maintain that our method generally produces good results with very low requirements, while simultaneously being controllable and editable.

2 Related Work

There are many facial animation techniques, including 2D image-based methods, and 3D geometry-based methods. A background on many of these techniques can be found in the survey by Parke and Waters [31]. However, the focus of this paper is on lip syncing to audio, which we summarize below.

2.1 Visual Speech Animation

The synthesis of realistic visual speech animations corresponding to novel text or prerecorded acoustic speech input has been a major research problem for decades. A common approach is to map one or more individual phonemes to corresponding viseme and generate the animation by interpolating the visemes given phoneme sequences [18] and demonstrated in a system using blendshapes [35]. However, naively interpolating viseme sequences tend to generate poor results since it does not consider co-articulation. Visual speech co-articulation is a phenomena describing that, which shows that a current viseme shape is not an independent face shape but rather would be affected by adjacent phonemes. The early work by Cohen et al introduced dominance functions [11, 28] as a parameterization method to deal with co-articulation. Their pioneering work is followed by more research works aimed to improve the dominance function model [13, 24, 12]. In their work each parameter curve controls a time-varying deformation over a small region on the face model. Although this parameterization method is intuitive to use, it is difficult to generalize across different faces since a set of parameters is bounded to a certain facial topology. Therefore a lot of manual effort is required for each new face. Our method parameterized the face animation as a set of blend curves using a canonical set of viseme shapes. Therefore the same set of parameter curve can be applied on different characters as far as they use the same canonical definition for viseme shapes.

Many data-driven methods are developed to learn co-articulation patterns from animation data. Ma et al learned variable length units based on a motion capture corpus [26]. Deng et al proposed a method that learn diphone and triphones model presented in weight curves from motion captured data [16]. Dynamic programming is applied to calculate the best co-articulation viseme sequences based on speech input. Although their method can produce natural speech animation with learned co-articulation, it requires large amounts of motion data for training and the result is highly dependent on the training data. Cao et al transferred captured data to a model to synthesize speech motions [6]. The recent work by Taylor et al [34] also introduced a dynamic visemes method, which extracts and clusters visual gestures from video input of human subjects. To synthesize dynamic visemes from a phoneme sequence, it looks for the best mapping through the probability graph learned from visual gesture data and stitch the viseme sequences together. Their method produces good results with co-articulation effects but requires significant animator efforts to setup around 150 visemes animations for each rigged characters face. By contrast, our method requires only a small set of static facial poses for each character.

Blendshape is a widely used technique due to its simplicity, when 3D shapes are interpolated and blended together in customized ways. One of the drawbacks when using blendshapes is the difficulty in defining a set of face shapes that are orthogonal to each other. This in turn causes the influence from one face shape to degrade other shapes. [25, 15] provided a method to avoid interferences. Human performance also provides a way to produce high quality and realistic facial animations. Waters and coworkers [36] applied radial laser scanner to capture facial geometry from the subject and used the scanned markers to drive the underlying muscle system. Chai et al [10] used optical flow tracking to capture the performance from a subject to drive facial animations.

2.2 Expressive Facial Animation

Expressive facial animations, including eye gaze and head movements, would greatly enhance the realism of a virtual character. Cassell et al developed rule based automatic system [8]. Brand et al [5] constructed a facial internal state machine driven by voice input using Hidden Markov Model (HMM). Deng et al extended the work in co-articulation model [16] and added the expressive facial motion learned from motion data to wrap it along with speech motion [17]. Cao et al used learning techniques to generate separate emotion components from speech [7]. More recent research has used test-to-speech(TTS) to drive expressive speech [29].

2.3 Software

FaceFX is a lip syncing software that has been widely adopted in many video games and simulations. It takes a speech audio as input and generate a set of blending curves. It then additively combines a set of simple component animations based on blending curves to produce facial animations. In this paper we will compare our result with its result based on lip-syncing accuracy and naturalness. In addition to prerecorded audio files, we also make use of text-to-speech (TTS) engine such as Microsoft TTS [22], Fesitival [4], and Cereproc [9] to produce mid-quality speech audio and phoneme timing on the fly.

Our method differs from many of the existing machine learning methods in that the data can be constructed and modified by

a skilled animator, and therefore does not require any performance capture or speech corpus. In addition, each phone bigram generated by the animator has local effects and can be edited or changed individually without affecting other parts of the system. Such process can be repeated for new characters, and thus the fact that our system can be changed and modified easily represents a strong advantage over machine learning methods which cannot learn and change the results for individual sounds per character as our system can.

3 METHOD

Our method involves two phases; an offline phase where an animator constructs animation curves associated with all pairwise combinations of phonemes, and a runtime phase that produce the speech animations by stitching, smoothing and constraint satisfaction based on the input timings of the phonemes.

3.1 Offline Phase

Our goal is to construct a set of animations that are associated with pairs of phonemes. We choose phoneme pairs as our canonical unit of animation since pairs of phonemes allow for coarticulation effects that are not possible by associating animation with individual phonemes. Diphones are commonly used during machine learning techniques, which represent timings between the middle of one phoneme and the next. Animators construct phone bigrams, which represent the timings from the start of one phoneme to the end of the following one. We choose phone bigrams, and not diphones, since an animation of a phone bigram is intuitive, whereas an animation of a diphone has no intuitive representation. Phone bigrams can be constructed by animating a short word with a single syllable that represents the two phonemes. For example, when animating the phone bigram for L-Oh, an animator can create an animation that represents the word 'Low'. By contrast, there is no intuitive representation of the mouth movements for a diphone, which would require the animator to start the motion from the middle of the 'L' sound to the middle of the 'O' sound. By using phone bigrams, animators are able to create better quality set of animations.

We also choose phoneme pairs, rather than combinations of three phonemes or higher order sequences in order to reduce the amount of data needed to be produced by animator. We expect that better results could be obtained by animating or learning combinations of three phonemes or even longer sequences. However, our goal is to identify a tractable set of data that can be quickly generated so that high quality lip syncing can be produced easily.

3.1.1 Phoneme Selection

Phonemes differ from each other according to their sounds. However, similar-looking facial movements can produce multiple phonemes. For example, a person will make a similar-looking face of folding their lips together when producing the 'b' and 'p' sounds. Thus, in order to further reduce the amount of data necessary for our method, we map the 40 English phonemes [37] into a smaller phoneme set, which we call the Common Phoneme Set, in turn greatly reducing the number of pairwise phonemes that must be produced during the offline phase. Table 1 shows the mapping to our common set of phonemes.

The mapping of phonemes (for example, mapping all 'b' and 'm' sounds to the common 'bmp' phoneme) somewhat reduces the subtleties that result from the differences between those sounds. However, our method allows for the inclusion of additional detail as needed by remapping the phonemes. Thus, the 'b' and 'm' phonemes could be separated, resulting in an additional number of phone bigrams to animate.

A language with n phonemes uses n^2 pairwise phoneme combinations. Thus, English has 40 phonemes and $40^2 = 1600$ pairwise phoneme combinations. By contrast, our reduced Common Phoneme Set of 21 phonemes contains $21^2 = 441$ pairwise phoneme combintions. Thus, animators only need to generate approximately 25% of the animations that would ordinarily be constructed if the entire English phoneme set were used.

English phoneme	Common Phoneme Set	Examples
ae, ah, ax	ah	cat, cut, ago
aa	aa	father
ao	ao	dog
ey, eh	eh	ate, pet
er	er	fur
ih, iy	ih	feel, fill, debit
w, uw, uh	w	with, too, book
ow	ow	go
aw	aw	foul
oy	oy	toy
ay	ay	bite
h	h	help
r	r	red
l	l	lid
s, z	z	sit, zap
sh, ch, jh, zh, y	sh	she, chin, joy, pleasure, yard
th, dh	th	thin, then
f, v	f	fork, vat
d, t, n, ng	d	dig, talk, no, sing
k, g	kg	cut, gut
p, b, m	bmp	put, big, mat

Table 1: Forty English phonemes mapped to our common set of phonemes. The left column lists the full set of English phonemes. The right column all the canonical visemes and the second right column lists all the pairwise phoneme combinations for the given phoneme schedules.

Using this reduced phoneme set, we generated phone bigrams from a corpus of approximately 200 utterances of varying length, from a single word, to utterances composed of several sentences. Table 2 shows the frequencies of the Common Phoneme Set pairwise phoneme combinations when generated from the Microsoft TTS Engine:

Note that the *d* Common Phoneme Set phoneme appears frequently, since it encompases four phonemes: *d*, *t*, *n*, *ng*. In addition, 104 of the 441 Common Set phone bigrams never appeared, and thus, nearly 99.1% of the phone bigrams could be generated from the first 283 most common phone bigrams. The phone bigram distribution is shown below in Figure 2. It is possible that different phoneme sequencers would generate slightly different results based on their own analysis of words and their mapping to the phonemes. However, we would expect them to be mostly aligned with our results, since to do otherwise would indicate a lack of synchronization with the SAMPA [37] phoneme set.

3.1.2 Facial Pose Selection and Phone Bigram Animation

Our method sequences a series of animations that are associated with our Common Phoneme Set of phoneme pair. Therefore instead of having animators create animations from scratch, we choose to represent phone bigram animations as a set of blend curves using a canonical set of face poses. This not only greatly simplifies the task for animators, but also allow our animations to be portable to different characters. As far as the new character has the same set of canonical face poses, our phone bigram curves can be applied on that character to generate high quality lip syncing animations. To

Rank	Common Set Phoneme Pair	Percentage
1	ah - d	4.62
2	d - ih	3.44
3	ih - d	2.68
4	d - d	2.64
5	d - ah	2.40
6	eh - d	1.83
7	d - w	1.66
8	ah - r	1.58
9	ih - w	1.50
10	d - z	1.43
11	ih - z	1.37
12	ih - kg	1.32
13	aa - d	1.27
14	z - ih	1.22
15	r - ih	1.21
16	th - ah	1.19
17	z - d	1.16
18	kg - ah	1.15
19	ah - l	1.10
20	z - ah	1.04
21	ah - bmp	0.98

Table 2: Frequency of Common Phoneme Set. Phoneme pairs generated from a TTS engine on a corpus of approximately 200 utterances.

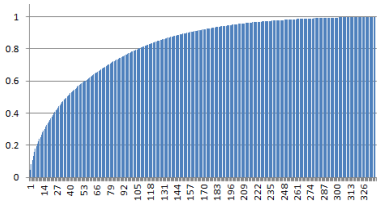

Figure 2: Distribution of Common Phoneme Set phoneme pairs using a text-to-speech engine. Horizontal axis shows the number of pairwise phonemes. Vertical axis shows the percentage of all pairwise phonemes.

produce a Common Phoneme Set of phone bigram animations, animators construct facial movements by combining several static face poses over time using a curve editor or similar tool. The animation curves are normalized, and will be timewarped over the length of the phone bigrams.

In order to reduce efforts from animators when creating blend curves, we develop a novel phone bigram curve editor which would assist the user to quickly create curves and evaluate the quality of resulting facial animations. As shown in Figure 3, our editing tool [32] allows the animator to choose pairwise phoneme combinations and edit their corresponding blend curves. The animator specifies a piecewise linear curve $l_j^k(t)$ for each face pose f_k and for each phoneme pair d_j. The linear curve represents the control curve for a smooth B-spline curve $c_j^k(t)$, which will be use to adjust the influence of pose f_k for a specific phone bigram d_j. Here each pose f_k represents displacements from neutral pose and therefore the weights $\sum_k c_j^k(t)$ does not necessarily sum to one. It can also generate a speech animation for a specific sentence automatically from a TTS engine or from a prerecorded audio file. The sentence will be analyzed on the fly and transformed into the individual phone bigrams. We show an example of a set of curves associated with the F-Ah phone bigram (for example, when saying the the beginning of the word 'fat'). After the animator updates the blend curves for a specific phone bigram, he can immediately evaluate its quality in a speech animation by typing words or sentences that contain the phoneme pair. This feedback can also help the animator to quickly identify the subset of phone bigrams that need improvements. By testing various words and sentences, problematic results can be effectively found and provide informations for the animator to fill in missing phone bigrams or improve existing ones.

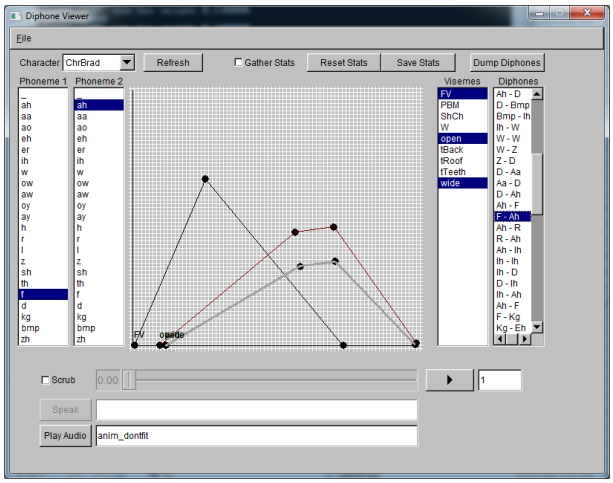

Figure 3: Curves for F-Ah phone bigram. The animator selects three facial poses; FV, open and wide, and constructs animation curves over normalized time. The animation can be directly played on the character, or the character's lip animation could be driven using TTS or recorded audio.

Since we rely on blend curves to produce character independent phone bigram animations, it is crucial to choose a good set of canonical face poses. In our method, we choose a set of facial poses that allows the generation of nearly any facial expression through various combinations of the facial poses. For compatibility with existing pipelines and for purposes of comparison, we chose the same facial poses that are used for the FaceFX software [20] and detailed in Figure 4. These shapes include 5 face shapes, and 3 tongue shapes. We expect that any canonical set of facial poses could be used, since the animator decides how and when to activate the various components. The facial poses need to be able to model sophisticated facial movements such as pursing the lips and tongue movements.

Figure 4: From first to the end: fv, open, pbm, shch, w, wide. In addition, three tongue positions: up, down and back are used as well as a neutral pose.

The animators construct curves in normalized time. The phone bigram animation will be used by the realtime algorithm by time-warping the animation curve over the length of each phone bigram, detailed in the section below. Each language, such as English, requires its own set of animations, although other language sets can reuse the same phone bigram animations, since many phonemes are shared between languages.

3.2 Runtime Phase

Figure 5 shows a flow chart for our runtime system. The runtime component operates on sequence of phonemes generated from a phoneme scheduler. The phoneme scheduler can be extracted directly from a TTS engine, extracted offline from recorded audio, or online via various phoneme translator tools. We have tested our TTS path using the Microsoft [30], Cereproc TTS [9], and Festival [3] TTS engines. Recorded audio can be extracted using various commercial [19] or noncommercial tools [23, 33]. The details of such phoneme scheduling are outside of the scope of this work. Our system expects a sequence of English phonemes and timings for each phoneme.

3.2.1 Curves Stitching and Smoothing

Our method then groups the sequence of phonemes into phoneme pairs, and maps those to the Common Phoneme Set. As shown in Figure 6, assuming the input phoneme schedules $p_0, p_1, ...p_n$ occur at times $t_0, t_1, ...t_n$. Our method first constructs a sequence of phoneme pairs consisting of adjacent pairs of phonemes $(p_0, p_1), (p_1, p_2), ...(p_{n-2}, p_{n-1})$ and their corresponding time span $(t_0, t_2), (t_1, t_3)..., (t_{n-2}, t_n)$.

Each phoneme pair (p_i, p_{i+1}) is also associated with a set of phone bigram curves $c_i^k(t)\{t_i \leq t \leq t_{i+2}\}$ for each canonical face pose f_k. We stretch or compress the curves according to the time span of the phoneme pair to generate $c_i^k(t)$. We then stitch the overlapping curves for the same face pose f_k over adjacent time spans to produce one single continuous blend curve $c^k(t)$ $\{t_0 \leq t \leq t_n\}$. The stitching process eliminates overlapping areas between adjacent curves c_i^k and c_{i+1}^k by retaining the curves with largest value, as shown in Figure 7. For example, $c^k(t) = max(c_i^k(t), c_{i+1}^k(t))\{t_{i+1} \leq t \leq t_{i+2}\}$. We choose the maximum value over linear blending or averaging in the overlapping area so that the original information is preserved in the resulting curves to activate face poses. Applying other blending methods may damp the curves and produce undesired lip movements during transition.

We also perform a smoothing pass over each curve using a user-specified window to scan through temporal domain and find local maximas. Figure 8 shows the resulting curves after the smoothing process. Here we defines a sliding window, with its size $2t_w$ defined by user. The smoothing window will be slideing over the curve $c^k(t)$ from t_0 to t_n and detect the local maximums for $c^k(t)$. If at any time instant t there are two or more local maximas $c^k(t_a)$ and $c^k(t_b)$ inside the window from $t - t_w$ to $t + t_w$, we will smooth out the curve values between t_a and t_b by interpolating two maximas $c^k(t_a)$ and $c^k(t_b)$ with a spline curve. Specifically, since $c^k(t_a)$ and $c^k(t_b)$ are the values of spline curves associated with piecewise linear curves whose values are $l^k(t_a)$ and $l^k(t_b)$ at time t_a and t_b, the new spline curve can be obtain by linearly interpolating $l^k(t_a)$ and $l^k(t_b)$ such that $l^k(t) = \frac{(t_b - t)l^k(t_a) + (t - t_a)l^k(t_b)}{t_b - t_a}$ from time t_a to t_b. This new linear curve thus defines a updated values for $c^k(t)$ from t_a to t_b. Intuitively, this process smoothes out the "valley" between two maximas and therefore helps reduce high frequency lip movements that are not natural for a speech animation.

3.2.2 Constraint Satisfaction

Since facial poses are activated based on parametric values, any two poses could be arbitrarily combined together to form a new face shape. However, certain poses, when activated simultaneously, would produce unnatural results. For example, the face pose for 'open' viseme should not be activated along with the pose for 'bmp'. Since one pose is open mouth and the other is close mouth, combining them together may cause interference and cancel each other. This can cause important information being eliminated from the resulting facial motion. To fix this problem, we added the constraints between face pose pair to filter out undesired poses. A constraint threshold $C(a,b)$ is defined for each pose pair (f_a, f_b). A priority value for both $P(a)$ and $P(b)$ is also defined for each face pose f_a and f_b to determine which pose should have its parametric value reduced during conflict. The threshold $C(a,b)$ determines whether the two poses f_a and f_b may interfere with each other, and it is activated when $c^a(t) + c^b(t) > C(a,b)$. Once the constraint is activated, the parametric value from the lower priority curve is reduced to satisfy the constraint $c^a\prime(t) + c^b\prime(t) \leq C(a,b)$. Assuming $P(a) > P(b)$, we define $c^a\prime(t) = c^a(t)$ and $c^b\prime(t) = c^b(t) - (C(a,b) - c^a(t) - c^b(t))$. Figure 9 shows the resulting curves after constraint adjustments. This simple heuristic ensures that the incompatible curves will not affect each other and thus it improves the quality and expression of resulting animations. The method is not restricted to a standard set of face poses, it can be extended to work with any arbitrary, non-conventional sets of face poses since the animator can define suitable thresholds and priority values based on the given face pose set. In our example, we found that setting the constraints $C(open, FV) = C(open, pbm) = C(open, ShCh) = 0.5$, $C(a,b) = 2.0$ for all other face pose pairs, and priorities $P(open) < P(W) < P(wide) < P(ShCh) < P(FV) < P(pbm)$ work well under standard FaceFx face pose set.

3.2.3 Parametric Speed Limits

To produce natural lip syncing animation, a character should only move his lips with a reasonable speed. However, the animator may create a bigram curve with high slope and the time warping could also compress the curve to be excessively sharp. This can cause fast activation or deactivation of various static face poses or shapes, resulting in popping or similar artifacts. We cap the parametric speed $\frac{dl(t)}{dt}$ of piecewise linear curve $l(t)$ by a value l_{max}. This prevents overly fast changes to any shape, and is intended to track the speed at which a person's face can be changed (for example, how quickly the mouth can be opened or closed). We have found that values where $l_{max} = 15$ provides reasonable results. Figure 10 shows the resulting curves after limiting the parametric speeds.

Once the speed limit phase has been completed, we transfer the curves to our animation system, which blends the facial poses or blendshapes according to their activation values dictated by the curves.

3.2.4 Editing

By constructing animation curves that are driven by a fixed set of shapes or poses, the animator constructs a data set that can be reused on other characters which use similar facial poses or blendshapes. Thus, during the offline phase, the animator creates character-independent animations. In addition, our method is well-suited for an animation pipeline since there are no black-box components whose results are difficult to interpret or modify. Specific segments of the animations can be directly mapped back to their originating phone bigrams and subsequently changed. In addition, character-specific animations can be added to the original set of phone bigrams curves to adjust the motions per character. Thus, our method allows two types of adjustments; changing the static poses

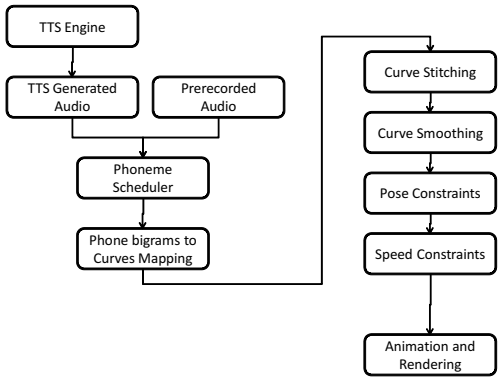

Figure 5: Runtime phase flow chart. A phoneme scheduler produces phonemes and their respective timings. Animations previously created by an artist are associated with each phone bigram animation. Each facial pose then stitches together those animations, runs a smoothing process, speed and pose constraints. The final curves are animated by the system.

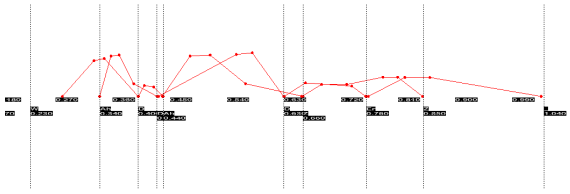

Figure 6: Curve visualization from our interactive tool. First step: Curves of the phone bigram animations from the 'open' viseme are shown and placed together on the timeline (shown in red). Phoneme boundaries are shown as vertical lines. Time axis is shown in the first row of the black boxes underneath the curves. Phonemes and their corresponding times are shown in the second and third row.

or blendshapes, as well as the changing of specific phone bigram animations on a per character basis.

4 Study

It can be difficult to evaluate the relative quality of various lip syncing algorithms against each other because of the difficulty in implementing each solution and the supporting data that is needed, such as character models, rigs and configuration parameters. Thus, most lip syncing algorithms are typically indirectly compared with each other taking into account those differing aspects. However, we are able to compare our results directly with a popular commercial lip syncing solution; FaceFX. For our experiment, we use the same character, the same phoneme scheduler, and we construct phone bigram animations using the set of FaceFX static poses. When producing results from our algorithm, the FaceFX phonemes are then mapped to our Common Phoneme Set. Thus our algorithm differs from the results in FaceFX only in the interpretation and translation of phonemes into animation curves. Therefore we can directly compare the two methods.

4.1 Method: Comparison With Other Methods

We performed a study comparing the results from both algorithms. The study includes 4 characters; one cartoon character, one high quality (near photo-realistic) character, one medium quality female

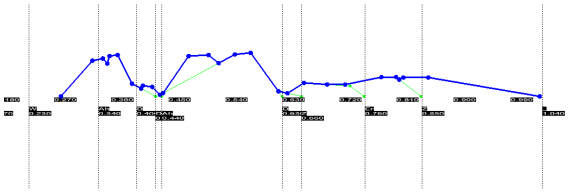

Figure 7: Curve visualization from our interactive tool. Second step: Overlapping curves are stitched together;parametric values of curves are compared in the overlapped regions and only the ones with largest values are retained (shown in blue). The original stitched curves from previous step are shown in green.

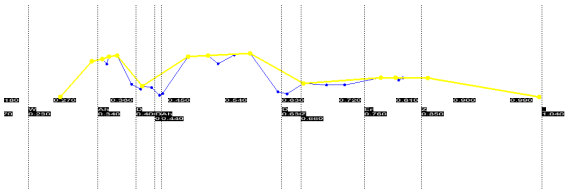

Figure 8: Curve visualization from our interactive tool. Third step: The stitched curves are smoothed by finding local maximums with a sliding window through temporal domain. The valleys of the curves are removed by connecting two local maximums inside the sliding window. The smoothed curves are shown in yellow.

character and one medium quality male character as shown in Figure 11. We captured 10 movie clip pairs for each character, each pair is one random utterance from the pool generated using our method and using FaceFX. A total of 80 people participated in the study with 20 observers assigned to each character's 10 movie clip pairs, so 200 choices are made for each character's lip animation performance. The positioning of the movie clip inside a pair is randomized.

The survey was done on Amazon Mechanical Turk [2] asking observers to choose the preferred clip between the paired clips as well as state the strength of their preference, using a scale from 1(weak) to 5(strong). Subjects were asked to choose based on the general performance of the lip animation accuracy and naturalness. Accuracy indicates that the mouth configuration is synchronized with the words heard from the audio. The higher the accuracy, the more you can understand what the character is saying by just reading the lip syncing even without sound. Naturalness indicates closely the mouth configuration resembles that of a human's during speech.

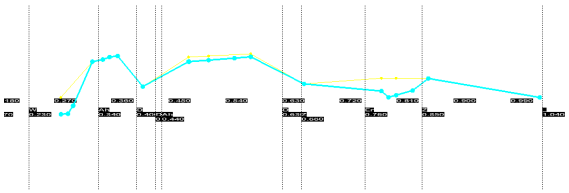

Figure 9: Curve visualization from our interactive tool. Fourth step: The curves are adjusted by enforcing the constraints between each face pose pairs. Conflicted face pose curves are adjusted according to their priority values. The resulting curves after adjustments are shown in cyan.

Figure 11: The four characters used in our study (top left) a cartoony character, (top right) a high quality character, (bottom left) a medium quality male character, (bottom right) a medium quality female character.

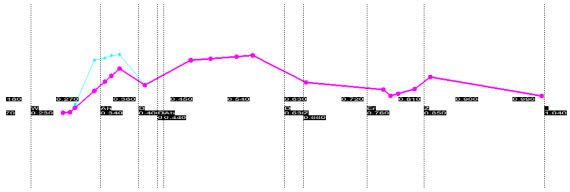

Figure 10: Curve visualization from our interactive tool. Fifth step: The curves are further adjusted to satisfy the speed limits. The slopes of curves are reduced if their parametric speeds are over a user-defined threshold. The final curves are shown in magneta.

4.2 Study Results

All the results are shown in Figure 12. Our method is preferred over FaceFX with 47%, 34%, 23%, 39% stronger preference respectively for cartoony character, high quality character, medium quality male character, medium quality female character and there's a significant different using chi-square goodness of fit test on preferences, $\chi^2_{(2)} = 44.14$, $p < .005$; $\chi^2_{(2)} = 23.12$, $p < .005$; $\chi^2_{(2)} = 10.58$, $p < .005$; $\chi^2_{(2)} = 30.42$, $p < .005$. When it comes to average strength of preferences, our method versus FaceFX is $4.03/3.70$, $3.97/3.80$, $3.54/3.60$, $3.62/3.7$ respectively.

4.3 Study Discussion

As expected, the results show our method is favored more by the participants across all types of characters. Participants that preferred our technique preferred it on average at greater than medium strength, with a preference strength that is greatest in the case of the cartoon character(4.03) and high quality character(3.97). Interestingly participants that preferred FaceFX, though fewer in number, also preferred it on average greater than medium strength. This suggests the need for a follow-on study where we tease apart what factors are leading to these judgments across techniques as well as across character types. Specifically in reference to our technology, there is a question of why the cartoony and high quality characters have stronger preferences than the medium quality. Is it just the quality of the character or is there an interaction between character quality and the realization of the visemes?

5 Discussion

In this paper, we present a lip syncing method that can achieve high-quality results using only artist-driven data. We observe that only phoneme pairs are needed, and that higher-order phoneme sequences are not necessary for generating reasonable synchronized lip movements with audio. We also observe that artist-designed animation curves work well at run-time to synthesize high quality speech animations, and that machine learning is not necessary. A key advantage to our method is that in can be constructed from any set of static facial poses or shapes. This allows the construction of lip animation for non-human characters using non-standard face poses, as well as allowing the construction of character face rigs separately from the animation, since the facial setup needed can be determined in advance, and does not have to be explicitly retargeted.

We also demonstrate that such a method can be effective on a

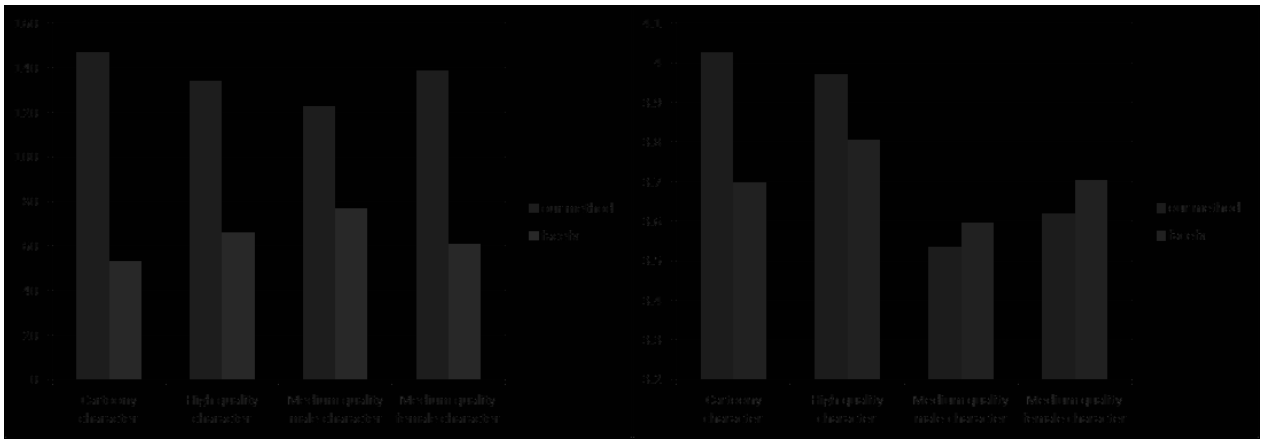

Figure 12: Comparison between our method and FaceFX using cartoony, high quality, medium quality characters. (Left) preference comparison (right) average strength of preference comparison. We use Amazon Mechanical Turk to collect viewer ratings from 80 participants.

range of character styles, including cartoon-like and realistic looking models. High levels of facial realism are becoming increasingly popular, and thus spurring the need for animation methods suitable for such levels of fidelity.

5.1 Data-Driven Methods

We observe that many previous methods attempt to solve the lip syncing problem by extracting co-articulation effects from captured data. The goal of these methods can be considered as generating a correct facial animation for each diphone or triphone combination through either heuristics such as dominance function or through machine learning methods. Therefore on a higher level, they attempt to achieve the similar goal as our method to fill out the phoneme pair or phoneme trio (diphones/phone bigrams, or triphones/phone trigrams) tables to account for co-articulations.

The limitation with data-driven methods is that there is no guarantee that all the required combinations exist or could be extracted effectively in the training data. Therefore the missing combinations would directly affect the quality of resulting animations. Moreover, there is no easy way to modified the learned model to account for missing combinations. The goal of our method is to provide a transparent framework for the animator to explicitly fill out the phoneme pair/diphone/phone bigram tables. Thus the animator has direct control over the quality of resulting speech animations. Note that although we did not choose to solve the problem via data-driven methods, our framework can be extended to make use of captured facial animations. Diphone curves, instead of phone bigrams, can be extracted from captured animations by fitting the animation data with canonical face poses.

5.2 Offline Data Generation

Our offline phase requires an animator to generate 441 short animations for the English language, which can be used by any character that utilizes the same speech targets used during the animation generation phase. Our animators were able to generate a single phone bigram animation in just a few minutes. Our supplemental video shows an example of generating such animations. Thus, if we conservatively estimate that each phone bigram animation takes 5 minutes to generate, an entire language set can be generated in approximately $441 * 5 = 2205$ minutes $= 37.5$ hours. Thus, a new language data set can be generated by a single animator using standard tools in less than 1 week. This language data set can be used for all characters which utilize the same static facial poses. In addition, we provide the English language data set openly to the community [1] so that others may use it directly in their experiments or works, thus further lowering the barrier to implementation for this method.

In addition, since nearly 25% of the Commone Phoneme Set phoneme pairs never appeared in our speech corpus, a mostly complete animation set in English can be generated in 75% of that time, or around 3 days. Short sentences can require around 30-40 unique phoneme pairs to be annotated, which typically take a only a few hours to construct. Thus, our lip sync method can be tested on entirely new 3D characters that have distinctive facial poses or shapes in a relatively short amount of time.

5.3 Configurability

A key advantage to our method is the configurability of the facial setup to the lip syncing setup. Any set of facial poses can be used to construct a language set which will be effective for all characters that use the same matching facial poses. Machine learning methods generally dictate the facial setup needed based on the learned data; for example, in generating shapes based on a statistical analysis such as PCA. Our method can adopt to any set of input data. This can be effective when the facial requirements have been dictated to the animator, such as when acquiring models from a 3D marketplace where the facial poses have alreay been established. This is also useful for non-human characters whose facial configurations either don't match a human's or contain a different set of poses that activate various non-human capabilities. With our method, it would be possible to construct a different phone bigram animation set to match any standard lip syncing configuration, thus allowing direct compatibility with other lip syncing setups.

[1] http://smartbody.ict.usc.edu

We purchases a rigged crab model from an online marketplace [1] which contained several non-standard facial poses, such as 'bare teeth', 'frown', 'smile', 'mouth open', 'mouth OO' and various tongue positions, such as those seen in Figure 13. We then constructed phone bigrams necessary to animate the crab, since our method allows the generation of such animation for use in lip synching from any set of facial poses. Our supplementary video shows the results. Note that we are not performing any remodeling or rigging changes in order to create the lip sync data set.

In addition, phoneme pairs can be individually identified and replaced as needed on a per character basis. These phoneme pairs are easy to identify, since their effect can be scene at the times corresponding to the phoneme activation. As mentioned above, this enables the partial construction of a lip sync animation set for testing purposes, without having to complete the entire set (for example, to preview the quality of the animation on a single sentence), since the parts of the lip sync that need to be consructed are readily indentifiable from their phoneme schedule.

Recent work by Taylor et al [34] have also utilized hand-crafted animations to generate lip sync data. Note that our method requires only a small set of static facial poses, while their method requires the construction of 150 short animations per character. We believe that the quality of our results are comparable to theirs while requiring over an order of magnitude less data per character.

5.4 Multilanguage Efforts

Our method can operate on multiple languages by creating a new set of animations based on the phonemes for the language or reusing existing phone bigram animations. We were able to generate a Mandarin phone bigram animation set by mapping the phonemes associated with Mandarin pinyin [38] to the our Common Phoneme Set in English, then adding one additional phoneme unique to Mandarin. The additional phoneme required adding 43 additional diphone animations, which were generated in less than one day. Thus our Mandarin lip synchronization could be generated in a short amount of time. We expect that other languages could be added through similar means.

5.5 Direct Method Comparisons

We were able to compare our method directly with FaceFX using a nearly identical pipeline. We do not claim that our method yields superior results to all other methods, but rather that our results are comparable to many others that have much larger requirements. Other methods that attempt to quantity the quality of animation are limited to learning based methods [27] or comparisons between the original data and the reconstructed data, which are not applicable to animator-driven methods. McGurk studies [14] can be performed to see how the McGurk effect on real humans compares to that of synthesized speech. Likewise, noise studies where words are randomly removed from utterances and must be recovered by reading lips can help judge the quality of the articulation. We believe that the best criteria for comparison is side-by-side comparison in subjective studies. We believe that our method is accessible and simple enough and adaptable to be used for comparison from future research.

5.6 Limitations

Our method does not address the issue of emotional content during speech. Many recent research works have identified expressive speech as a more interesting problem than simple lip animation, and thus have avoided the problem of generating high quality lip animation on its own by pursuing high quality emotional facial performances, presuming that a high quality emotional facial performance would include a high quality lip sync result. In doing so, however, there has been no consensus among the research community regarding the best lip sync methods for 3D characters. Lip animation independent of facial expression remains an important component of many games and simulations, as well as for methods that use the upper face as a means to express emotional content.

We anticipate that the use of phone trigram or triphones (or longer sequences of phonemes) would result in slightly better results, but would require too much data to be annotated by an animator. Triphones would require p^3 animations, where p is the number of phonemes in our Common Phoneme Set. Thus, manually annotating phone trigrams or triphones would make our method unwieldly. Also, slightly better results could be obtained by expanding the Common Phoneme Set. For example, separating the 'b' and 'm' phonemes. Our selection of the Common Phoneme Set is based on general practices and expert knowledge within the animation community.

5.7 On Achieving Good Lip Synching

There are many factors that contribute towards high quality lip syncing, including the model rendering, facial setup including rigging, and input data. Proper phoneme alignment is critical to generating properly timed lip syncing, which is why text-to-speech algorithms or slow, even-toned utterances can generate good results, while fast talking, emotional speech, stutters, partial word expressions and other non-words can be difficult to map properly. Research has show that the best phoneme alignment can be determined only as a margin of error by agreement among experts [21].

Acknowledgements

Thanks to Teresa Dey for her extensive artistic contributions in support of this work. Thanks also to Matt Liewer, Joe Yip and Oleg Alexander for artistic and technical support. Thanks to Jarvis McGee for his work on the curve editor. Thanks to Dominic Massaro for discussions about lip animation and dominance function methods.

References

[1] Daz3d, content marketplace, http://www.daz3d.com, 2013.

[2] Amazon. Amazon mechanical turk, 2012.

[3] A. Black, P. Taylor, and R. Caley. The festival speech synthesis system, 1998.

[4] A. W. Black and P. A. Taylor. The Festival Speech Synthesis System: System documentation. Technical Report HCRC/TR-83, Human Communciation Research Centre, University of Edinburgh, Scotland, UK, 1997. Avaliable at http://www.cstr.ed.ac.uk/projects/festival.html.

[5] M. Brand. Voice puppetry. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 21–28, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[6] Y. Cao, P. Faloutsos, E. Kohler, and F. Pighin. Real-time speech motion synthesis from recorded motions. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 345–353. Eurographics Association, 2004.

[7] Y. Cao, P. Faloutsos, and F. Pighin. Unsupervised learning for speech motion editing. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 225–231. Eurographics Association, 2003.

[8] J. Cassell, C. Pelachaud, N. Badler, M. Steedman, B. Achorn, B. Douville, S. Prevost, and M. Stone. Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. pages 413–420, 1994.

[9] Cereproc. Cerevoice text-to-speech synthesis, http://www.cereproc.com, 2012.

[10] J.-x. Chai, J. Xiao, and J. Hodgins. Vision-based control of 3d facial animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 193–206. Eurographics Association, 2003.

Figure 13: Some face poses from cartoon crab acquired from an online marketplace. Our method is able to use any set of facial poses to construct a lip animation data set.

[11] M. M. Cohen and D. W. Massaro. Modeling coarticulation in synthetic visual speech. In *Models and Techniques in Computer Animation*, pages 139–156. Springer-Verlag, 1993.

[12] M. M. Cohen, D. W. Massaro, and R. Clark. Training a talking head. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, ICMI '02, pages 499–, Washington, DC, USA, 2002. IEEE Computer Society.

[13] P. Cosi, E. Caldognetto, G. Perin, and C. Zmarich. Labial coarticulation modeling for realistic facial animation. In *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, pages 505 – 510, 2002.

[14] D. Cosker, S. Paddock, D. Marshall, P. L. Rosin, and S. Rushton. Toward perceptually realistic talking heads: Models, methods, and mcgurk. *ACM Transactions on Applied Perception (TAP)*, 2(3):270–285, 2005.

[15] Z. Deng, P.-Y. Chiang, P. Fox, and U. Neumann. Animating blendshape faces by cross-mapping motion capture data. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, I3D '06, pages 43–48, New York, NY, USA, 2006. ACM.

[16] Z. Deng, J. Lewis, and U. Neumann. Synthesizing speech animation by learning compact speech co-articulation models. In *Computer Graphics International 2005*, pages 19 – 25, june 2005.

[17] Z. Deng, I. C. Society, T. yong Kim, M. Bulut, S. Narayanan, and S. Member. Expressive facial animation synthesis by learning speech co-articulation and expression. *Space, IEEE Transaction on Visualization and Computer Graphics*, 12:2006, 2006.

[18] T. Ezzat and T. Poggio. Miketalk: A talking facial display based on morphing visemes. In *In Proceedings of the Computer Animation Conference*, pages 96–102, 1998.

[19] FaceFX. Facefx software, http://www.facefx.com/, 2012.

[20] FaceFX. Facefx speech targets, http://www.facefx.com/documentation/2010/w76, 2012.

[21] J.-P. Hosom. Speaker-independent phoneme alignment using transition-dependent states. *Speech Communication*, 51(4):352–368, 2009.

[22] X. Huang, A. Acero, H. Hon, Y. Ju, J. Liu, S. Meredith, and M. Plumpe. Recent improvements on microsoft's trainable text-to-speech system-whistler. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 2, pages 959–962 vol.2, 1997.

[23] D. Huggins-Daines, M. Kumar, A. Chan, A. Black, M. Ravishankar, and A. Rudnicky. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, pages I–I. IEEE, 2006.

[24] S. A. King and R. E. Parent. Creating speech-synchronized animation. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):341–352, may 2005.

[25] J. P. Lewis, J. Mooser, Z. Deng, and U. Neumann. Reducing blendshape interference by selected motion attenuation. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, I3D '05, pages 25–29, New York, NY, USA, 2005. ACM.

[26] J. Ma, R. Cole, B. L. Pellom, W. Ward, and B. Wise. Accurate Visible Speech Synthesis Based on Concatenating Variable Length Motion Capture Data. *IEEE Transactions on Visualization and Computer Graphics*, 12:266–276, 2006.

[27] X. Ma and Z. Deng. A statistical quality model for data-driven speech animation. *IEEE Trans. Vis. Comput. Graph.*, 18(11):1915–1927, 2012.

[28] D. W. Massaro, M. M. Cohen, M. Tabain, J. Beskow, and R. Clark. Animated speech: research progress and applications. pages 309–345, 2012.

[29] W. Mattheyses, L. Latacz, and W. Verhelst. Comprehensive many-to-many phoneme-to-viseme mapping and its application for concatenative visual speech synthesis. *Speech Communication*, 2013.

[30] Microsoft. Microsoft english phonemes, http://msdn.microsoft.com/en-us/library/ms717239, 2012.

[31] F. I. Parke and K. Waters. *Computer Facial Animation*. AK Peters Ltd, second edition, 2008.

[32] A. Shapiro. Building a character animation system. *Motion in Games*, pages 98–109, 2011.

[33] S. Sutton, R. Cole, J. D. Villiers, J. Schalkwyk, P. Vermeulen, M. Macon, Y. Yan, E. Kaiser, B. Rundle, K. Shobaki, P. Hosom, A. Kain, Johan, J. Wouters, D. Massaro, and M. Cohen. Universal speech tools: The cslu toolkit. In *In Proceedings of the International Conference on Spoken Language Processing (ICSLP*, pages 3221–3224, 1998.

[34] S. L. Taylor, M. Mahler, B.-J. Theobald, and I. Matthews. Dynamic units of visual speech. In *Proceedings of the 2012 ACM SIGGRAPH/Eurographics symposium on Computer animation*, jul 2012.

[35] A. Wang, M. Emmi, and P. Faloutsos. Assembling an expressive facial animation system. In *Proceedings of the 2007 ACM SIGGRAPH symposium on Video games*, pages 21–26. ACM, 2007.

[36] K. Waters and D. Terzopoulos. Modeling and animating faces using scanned data. *The Journal of Visualization and Computer Animation*, 2(4):123–128, 1991.

[37] J. Wells et al. Sampa computer readable phonetic alphabet. *Handbook of standards and resources for spoken language systems*, 4, 1997.

[38] P. H. Zein. Mandarin chinese phonetics, http://www.zein.se/patrick/chinen8p.html, 2012.