

Speech Driven MPEG-4 Facial Animation for Turkish

Arman Savran (1), Levent M. Arslan (2), Lale Akarun (3)

(1) Bogazici University Multimedia Laboratory (BUMM), Istanbul, Turkey

arman.savran@boun.edu.tr

(2) Bogazici University Multimedia Laboratory (BUMM), Istanbul, Turkey

arslanle@boun.edu.tr

(3) Bogazici University Multimedia Laboratory (BUMM), Istanbul, Turkey

akarun@boun.edu.tr

Abstract

In this study, a system, that generates visual speech by synthesizing 3D face points, has been implemented. The synthesized face points drive MPEG-4 facial animation. To produce realistic and natural speech animation, a codebook based technique, which is trained with audio-visual data from a speaker, was employed. An audio-visual speech database was created using a 3D facial motion capture system that was developed for this study. To improve the performance of the system when used by different speakers, a further training was performed with audio-only data from a small number of speakers. The resulting system is capable of animating faces from an input speech of any Turkish speaker.

1. Introduction

The synthesis of visual speech animation in computers has an important role for human-machine interaction. For instance, hearing-impaired can benefit from synthetically generated talking faces by means of lip reading. Lip-reading tutors for hearing-impaired, or language tutors for the children that have difficulties in speaking can be prepared. Also, videophones can be produced to make possible the distant communication of the deaf. Talking faces that are driven by a speech synthesizer can be employed as user interface agents, for example in e-learning, in Web navigation or as virtual secretary.

Unfortunately, realistically animating a face is a challenging problem, since we are all experts in perceiving facial cues and can not tolerate unnatural looking details. Inaccurate synthesis of the visual speech not only disturbs listeners, but also degrades the perception of speech. McGurk and McDonald [1] have proved that speech is perceived with both audio and visual signals together. They have noticed that when the auditory /ga/ was combined with visual /ba/ it was perceived as /da/. This is known as McGurk-effect, and it emphasizes the need for the accurate synchronization between visual and acoustic speech.

Various methods have been developed to animate a face using speech. Beskow [2] employed a rule based approach to synthesize articulatory parameters that control a 3D face model. Cohen and Massaro [3] used

dominance functions to model coarticulation, the influence of neighboring segments of a phoneme, to synthesize visual speech. In Video Rewrite [4], dynamic context dependent units of visible speech are concatenated to create speech video of a person. These techniques are called phoneme-driven articulation, since the visible articulatory movements are produced using a sequence of time-labeled phonemes.

On the other hand, in audio-driven articulation, audio signal directly drive the articulation. Lewis and Parke [5] classified acoustic signals into visemes, the visible speech units, using LPC spectrum. Yamamoto et al. [6] employed HMM-Viterbi and HMM-EM methods to synthesize visual parameters of lip movements from audio signal. In many works [7], Time-Delayed Neural Networks were used to map acoustic features to visual features by using succeeding and previous audio frames as well as the current frame in order to consider context information.

In this study, an audio-driven method is realized to drive an MPEG-4 compliant facial animation. This technique uses audio-visual codebooks as proposed by [8]. In this approach, articulation of the speech is synthesized based on real data gathered from a speaker. To capture this data, a 3D facial motion capture system, which tracks facial markers, was developed. This codebook based method has the advantage of capturing coarticulation effects that are crucial for realistic and natural facial animation. However, with this technique, the performance of the speech animation is degraded when used with different speakers, since the system is trained with audio-visual data from a single speaker. In this study, to make the system speaker independent, a new codebook is created by using audio-only data from a small number of speakers. Our system was trained on Turkish talkers, and thus is capable of creating speech animations for Turkish speakers.

The paper is organized as follows. In section 2, the facial motion capture system will be described. The method that synthesizes the face points to control a face model will be explained in section 3. In section 4, the MPEG-4 facial animation will be explained. The results of the experiments and evaluations for our system will be discussed in section 5. Finally, conclusion will be given in section 6.

2. 3D Facial Motion Capture

In order to capture the facial movements of a speaker, a 3D facial motion capture system has been developed for this study. It is an optical system which tracks the 3D coordinates of markers placed on the face. In this system, a stereo camera (Bumblebee Stereo Vision Camera [9]), which contains two CCD sensors, is used. To stream the stereo video at 30 frames per second and audio synchronously to the disk, recorder software was developed. The video is composed of 24 bit color images with 640x480 resolution. As markers, small circular color stickers are used. The block diagram of our 3D facial motion capture system is depicted in Figure 1. In this system, first, the facial markers are located from the initial frames of the video. Next, markers are tracked in 2D, and 3D coordinates are reconstructed from stereo. In the last step, global head motion is estimated with the help of markers attached on the constant parts of the face. The final coordinates are obtained after removing this estimated head motion from the tracked data. The details of the developed system is described in [10].

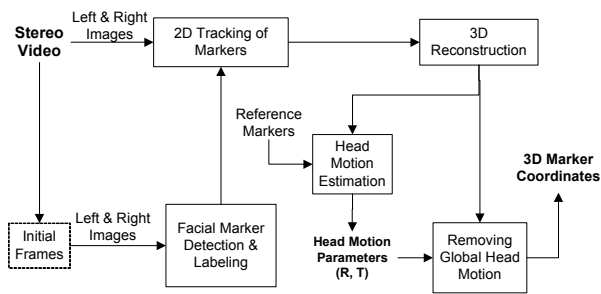


Figure 1 Block diagram of the 3D Facial Motion Capture System

In this system, in all image processing routines for detection and tracking, hue component of HSI (Hue-Saturation-Intensity) color space [11] is used. Since hue component is related with intrinsic character of a surface, lighting is normalized. To find the positions of facial markers on the initial frames, first, a face mask is extracted from the input image. Performing the marker detection using this face mask prevents confusions, when background pixel colors are closer to marker's color. It is obtained by finding skin-tone regions with thresholding and applying some binary image operations like connected component labeling and morphological operations. Having found the face mask, markers on this face are detected. These markers are circular stickers with blue and green colors. To detect these markers, first, thresholding is applied and binary images are generated. Then, after applying connected component labeling and area thresholding candidate markers are located by calculating the object centroids. However, some of these candidates may not be the markers. Therefore, the false candidates are eliminated using a circularity measure which is computed by using the radial distance of the objects [12].

However, the detected markers of left and right images have no correspondence, and to match them a labeling needs to be realized. This labeling is done automatically by using configuration information of the markers. Figure 2 shows an example result of the marker detection and labeling algorithms.

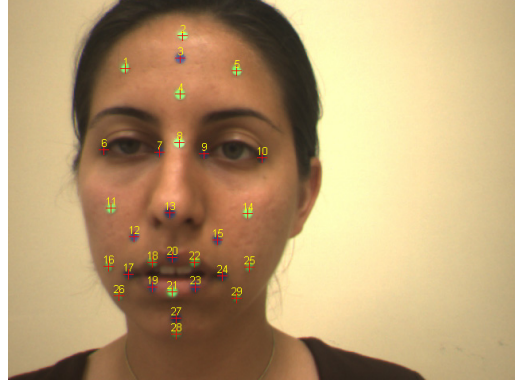


Figure 2. Detected and labeled markers

Tracking of markers from video images is performed with an algorithm which inherently considers color and shape information. In this algorithm, each marker's new position at the current frame is searched inside the square window centered at the previous coordinates. The algorithm is based on template matching with normalized cross correlation. For each marker, template and search images are obtained according to marker's color. Then, before the template matching, contrast and edge enhancement is applied to emphasize the marker's shape. Also, thresholding is employed only to consider reliable pixels for the template matching. Final coordinates of the tracked markers are obtained by calculating the centroid.

In the next step, 3D coordinates of the 2D tracked markers are reconstructed by using stereo vision. However, since humans move their heads unconsciously during speaking, the resulting facial motion is not only due to speech production but also due to the head movements. In order to take only the speech related motions of the facial points into account, the global head motions are estimated and removed. For estimation of the head pose, some markers are placed on the constant parts of the face so that we can treat head as a rigid object. The movement of a rigid object can be determined by a rotation matrix R , and translation vector T .

$$p_i = Rr_i + T \quad (1)$$

where p_i and r_i are the 3D positions of point i in the current frame and reference frame respectively. With respect to a reference, the relative movement of the head is predicted with the method proposed by Arun et. al [13]. The method is based on minimizing the least-squares error to estimate head rotation R and head translation T . After the motion of the head is determined, 3D facial points are normalized by the inverse transformation of Eq. (1).

3. Face Point Trajectory Synthesis

The articulation of visual speech is controlled by synthesized face point trajectories. To synthesize these trajectories, a method based on the technique proposed by [8] is used. The block diagram of the system is depicted in Figure 3. This technique employs audio-visual codebooks that store acoustic and visual features with associated context-phones. Also, a visual phoneme similarity matrix is created in the training phase in order to synthesize context-phones that are not available in the codebook. However, this technique is speaker dependent since training is performed with a single speaker. In this study, to make the system speaker independent the single speaker codebook is extended by appending audio features from different speakers. As acoustic features Line Spectral Frequencies (LSFs), and as visual features Principle Components of the measured 3D displacements are used. In the synthesis phase, facial trajectories are produced from incoming acoustic speech signal and accompanying phones by the weighted sum of selected codebook visual features. The final trajectories are obtained after smoothing the visual features and calculating the 3D face point displacements.

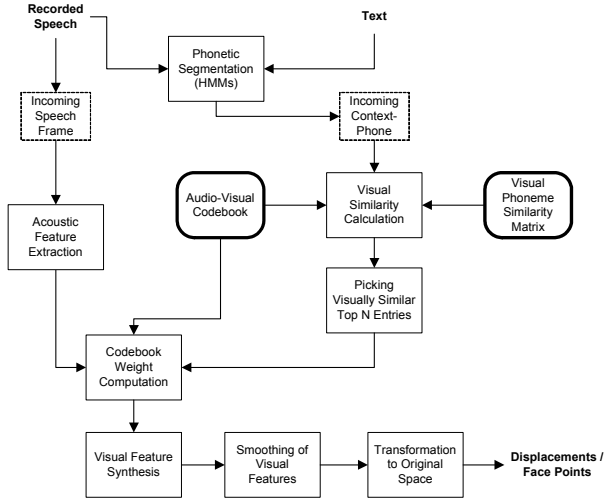


Figure 3 Block diagram of the codebook based face point trajectory synthesis algorithm

3.1. Visual Features

To represent facial dynamics, principle components (PCs) of the face point data are used. Principle Component Analysis (PCA) is used to map data to a new space where the dimensions (PCs) are uncorrelated and represent direction of maximum variance in the original data. Therefore, applying PCA to facial point data is very appropriate, because, the motions of these points are highly correlated.

In this work, first, a reference frame that has the neutral facial expression was selected from the dataset, and displacements of 3D points relative to this reference was calculated for the whole dataset. Then, PCA was

performed on this displacement data. This data were obtained from 3D positions of 29 facial markers for 200 utterances. This yielded 17802 vectors with 87 dimensions as inputs to the PCA. It was observed that first five PCs explained %90 of total variance in the data. Since there are few basic facial movements like mouth opening or lip protrusion for speech articulation, and the facial points are highly correlated, this PCA result is as expected. However, 20 PCs, which explain about %99 of variance, are used as visual features, since the subtle details which are necessary for more natural animation can be hidden in high ordered PCs.

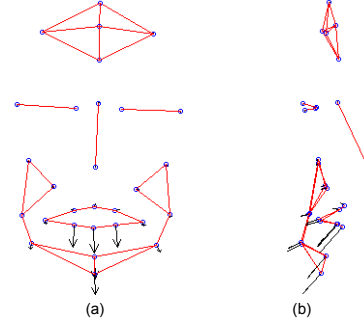


Figure 4 The influence of modification of first principle component ((a) frontal and (b) profile views)

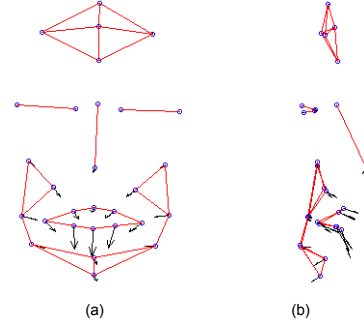


Figure 5 The influence of modification of second principle component ((a) frontal and (b) profile views)

In Figure 4, the contribution of the first principle component to the facial motion is depicted by front and profile views of the face. It was obtained by reconstructing the 3D displacements by only using the first PC. As it is seen from the figure, the first PC corresponds to the lower jaw movement, which is the most important facial movement during speech production and has the highest correlated facial points. This explains why its contribution to the variance is very high, above %60. Also it was observed that the second PC has the effect of lip protrusion (Figure 5), the third and fourth PCs are related to mouth opening and closing without jaw movement, and the fifth PC has the influence of labiodental occlusion. These first five PCs explain the most crucial movements for speech articulation. It becomes difficult to interpret facial

motions captured by high order PCs. However, their influence should be considered since they can encode the subtle details which are necessary for perceptual naturalness. For this reason, first 20 PCs are used as visual features in this work.

3.2. Acoustic features

In this face point synthesis procedure, Line Spectral Frequencies (LSFs) are employed to obtain useful vocal data. LSFs have close relationship with the speech formants, which are mostly determined by vocal-tract shape, and therefore they are convenient features for the estimation of face shapes.

In this study, the audio signals were digitized by 16 bits with a sampling frequency of 16 kHz. Before the feature extraction, a pre-emphasis filter ($H(z)=1-\alpha z^{-1}$; $\alpha=0.97$) is applied to slightly boost the higher frequencies. Then, linear prediction (LP) analysis of order 18 was performed for each 25 ms audio frame after windowing with Hamming window [14]. Finally, LSFs were computed from the LP coefficients with the methods described in [15, 16].

3.3. Audio-visual codebook

Audio-visual codebook generation is the main step of the training phase of the algorithm. This codebook contains audio and visual features with context information. Audio-visual codebook entries are created as follows. Figure 6 depicts the creation of a codebook entry for a representative context-phone $/l_2_l_1_c_r_1_r_2/$. First, speech signal is segmented into phonemes. In order to segment the speech signal, a hidden Markov model (HMM) speech recognizer run in forced-alignment mode is used. For the recognition task, Mel-Frequency Cepstrum Coefficients [17] are extracted from the audio signal. There are 29 Turkish phonemes trained for HMM recognition. These phonemes were modeled with three states and mixture of six Gaussians. Also, one short pause model with one state, and one silence model with three states were trained. Training was performed on a large dataset composed of Turkish speakers.

Having obtained the phonetic segmentations with the HMM recognizer, for each phoneme a context-phone is formed by combining preceding and succeeding phonemes with the current phoneme (Figure 6). In our training corpus total number of the context-phones is 4057. Then, for each context-phone, a codebook entry is created by adding audio and visual features. These features are extracted from the uniformly spaced time locations within the phoneme duration. In this process, visual features at these locations are obtained by linear interpolation. However, for the silence regions, corresponding visual features are not included in the codebook entry. Instead, these features, which are the weights of the PCs, are set to zero in order to prevent noise emerging at the silence regions by synthesizing neutral face shapes.

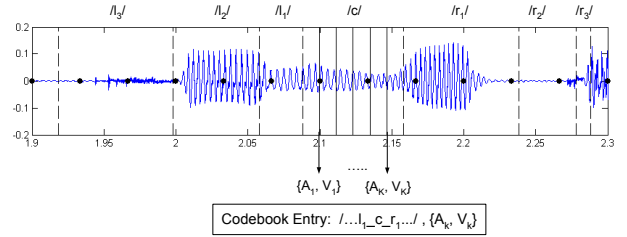


Figure 6 Demonstration of the creation of a codebook entry. Black dots represent the position of video frames. A_k and V_k are the audio and visual feature vectors respectively.

The audio-visual features in this codebook belong to a single speaker, and hence the synthesis for other speakers is not expected to work well. In this study, a new multi-speaker codebook was created to increase the synthesis performance when used by different speakers. The idea is to enrich the acoustic features for each codebook entry. Thus, a speaker-independent system is obtained by including different speaker characteristics in the codebook. For this purpose, audio-only training data with the same corpus, which is obtained from eight male and eight female speakers, was used.

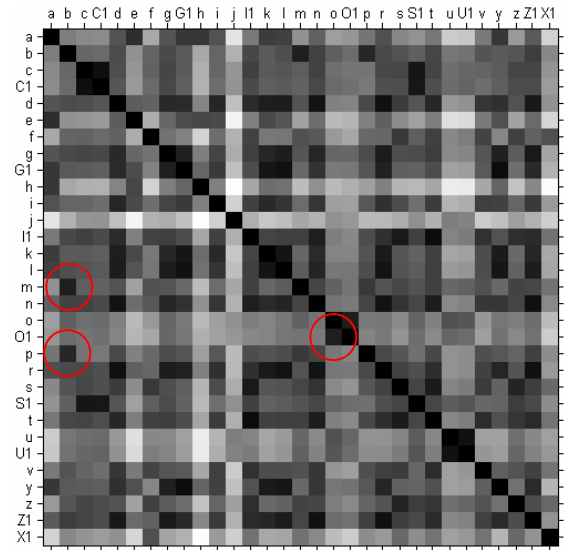


Figure 7 The derived visual phoneme similarity matrix for Turkish. Circles mark the similarity between phonemes $/b/$ & $/p/$ & $/m/$, and $/o/$ & $/O1/$.

3.4. Visual Phoneme Similarity Matrix

The second step in the training phase is generation of visual phoneme similarity matrix. This matrix is used during synthesis to cope with the incoming context-phones that are not available in the codebook, since having all context-phones in the training corpus is not possible. To create the matrix, visual similarity values of phonemes are calculated with the Euclidean distance of the principal components of face data. Hence, first, an average principle coefficient component vector is calculated for each phoneme. Then, the similarity measure is found by the below formula.

$$S_{ik} = e^{-v \cdot \|m_i - m_k\|}, \quad i = 1, \dots, K, \quad k = 1, \dots, K \quad (2)$$

According to this formula, similarity value for i th and k th phoneme S_{ik} will be between 0 and 1. Here, K is the number of the phonemes, and the constant v is to control the dynamic range of similarity values, and it was chosen as 10 for this study.

Figure 7 visualize the similarity matrix for Turkish phonemes as a gray-scale image, where the darker squares represent higher similarity values. In this figure, Z1 is the short pause, and X1 is the silence. It is seen that estimated similarity values are consistent with the expectations. For instance, from the Figure 7 it is seen that, the closest phonemes to /o/ is /O1/ due to lip rounding, and to /b/ is /p/ and /m/ due to lip closure.

3.5. Trajectory Synthesis

In the synthesis phase, PC vectors for the input speech are generated by linear combination of codebook PCs. However, only the codebook entries which have similar contexts in terms of face shapes to the input are used in this process. Therefore, the first step is to find the right codebook entries.

In order to find the entries, the context-phone of the incoming speech frame is compared to available context-phones in the codebook by the formulation given below.

$$n_j = S_{c_j} + \sum_{i=1}^C w^{-i} S_{l_{i,j}} + \sum_{i=1}^C w^{-i} S_{r_{i,j}}, \quad j = 1, \dots, L \quad (3)$$

In this formulation, S_{c_j} , $S_{l_{i,j}}$ and $S_{r_{i,j}}$ are the visual similarity values found in the visual phoneme similarity matrix; C is the level of context information; L is the total number of context-phones in the codebook; w is the weighting constant; and n_j is the score for the j th codebook entry. The subscripts of the similarity values denote the phonemes of the context-phones. In this work, w is set to 10, in order to make influence of center phonemes in the decision procedure always higher than the outer ones. After calculating the similarity scores, the top N most similar context-phones are selected from the codebook.

Having obtained the top N context-phones, the next step is to assign weights for them. For this purpose, first, distances between acoustic features of the input speech frame and the codebook features are calculated. The distance calculation makes use of important characteristics of LSFs to model the human perception of acoustic signals. Human hearing is more sensitive to formants that have narrow bandwidth. Since the LSF pairs indicate formants, and their distances are correlated with the bandwidths of corresponding formants, a perceptual weighting is applied during distance calculation by assigning higher weights to closely spaced LSFs,

$$d_i = \sum_{k=1}^P h_k \cdot |w_k - L_{ik}|, \quad i = 1, \dots, k \times N \quad (4)$$

$$h_k = \frac{1}{\arg \min(|w_k - w_{k-1}|, |w_k - w_{k+1}|)}, \quad k = 1, \dots, P \quad (5)$$

In the above expressions, w_k s and L_{ik} s are the LSFs of the input and the codebook respectively, P is the LSF vector dimension, and K is the total number of audio feature vectors found in the codebook entry. From these distances, a set of weights, v , are derived,

$$v_i = \frac{e^{-\gamma d_i}}{\sum_{l=1}^{K \times N} e^{-\gamma d_l}}, \quad i = 1, \dots, K \times N \quad (6)$$

These normalized codebook weights are used to approximate the original LSF vector w by linear combination of codebook LSF vectors,

$$\hat{w}_k = \sum_{i=1}^{K \times N} v_i \cdot w_{ik} \quad (7)$$

To determine the value of γ in the Eq. (6) an incremental search is employed. This search is done to minimize perceptual weighted distance between the original and the approximated LSF vector. The search range is [0.2, 2] in this study. Finally, the new PCs for the current speech frame are created with the weights v which represent the acoustic similarity,

$$\hat{p}(t) = \sum_{i=1}^{K \times N} v_i P_i \quad (8)$$

3.6. Smoothing of Synthesized Trajectories

The synthesized PC trajectories introduce noise that make the resulting animation unnatural, although the trajectories pass through proper regions. Therefore, smoothing must be applied. In this study, only the speech regions are smoothed, because silence regions are always on the baseline due to the zero magnitude PCs stored in the codebook. However, this causes jumps between silence and speech regions which are disturbing. To provide smooth transitions for these locations, hamming function is used after smoothing the speech regions. The process is a simple extrapolation realized by pasting the proper half of the hamming windows to the speech boundaries. By observation, the width of this half window was decided as 200 ms.

For smoothing the trajectories over the speech regions, ‘‘Least Squares Spline Approximation’’ technique [18] is employed in this study. This technique is a kind of data compression method, since the original signal is approximated with fewer number of spline coefficients. However, here, it is used to reduce noise present in the synthesized PC trajectories. The basic idea is the decimation of the spline coefficients, considering the approximation with minimum error. The expression for the spline approximation with an up-sampling factor m is

$$g_m^n(x) = \sum_{i=-\infty}^{\infty} y(i) \cdot \beta^n(x/m - i) \quad (9)$$

where the $y(i)$ s are the spline coefficients weighting expanded and shifted basis functions β^i in order to obtain approximated signal $g_m^n(x)$. For our smoothing purpose, cubic splines ($n=3$) with sampling factor $m=3$ is used.

After the smoothing, the final operation is to get the displacement data by transforming PCs to the original space.

4. MPEG-4 Facial Animation

Facial animation is generated by deforming a 3D polygonal mesh of the face model. The animation is driven by 3D face point trajectories through the MPEG-4 Facial Animation Standard [19]. To see the synthesis results, an MPEG-4 compliant facial animation engine [20] was used in this study.

According to MPEG-4 Facial Animation Standard, human head is parameterized with two parameter groups as face definition parameters (FDPs), and facial animation parameters (FAPs). FDPs define the appearance of the face, and include 84 feature points like the tip of the nose, mouth corners, etc.

The face animation in MPEG-4 is governed by FAPs. They indicate facial movements with respect to neutral face. There are 68 FAPs divided into 10 groups. The first two groups are the high-level FAPs which represents visemes and the most common facial expressions. On the other hand, the remaining FAPs allow more detailed animations by deforming local regions on the face. There are 66 low-level parameters categorized into groups corresponding to sub facial regions like eyes, lip, etc. The value of a low-level FAP determines the displacement amount for one Cartesian coordinate of the feature point due to translation or rotation. Therefore, FAP decoder is responsible for calculating the movements of other coordinates, and also movements of vertices in the proximity of that feature point. FAP values are expressed in terms of Facial Animation Parameter Units (FAPUs). These units are the fractions of key facial distances, except for FAPU used to measure rotations. The normalization of facial displacements by means of FAPUs allows animating any MPEG-4 face model in a consistent way.

In this study, to animate MPEG-4 face models, some markers were positioned according to MPEG-4 specifications. Totally, reconstructed coordinates of 15 markers were used for the extraction of FAPs. These markers were attached on the MPEG-4 feature point positions at the outer lips, at the cheeks, at the jaw, and at the nose. FAPUs were manually measured from the reference frame that has the face in neutral state. According to MPEG-4 FAP specifications, all of the related FAPs (total 20) are calculated for the animation.

To assess performance of the synthesis results, the Facial Animation Engine (FAE) developed at the University of Genova [20] was used. The FAE is a high-level interface for animating MPEG-4 compliant faces, and can provide high frame rate animation synchronous

with audio. This software can animate different MPEG-4 face models and reshape the existing ones with the FDPs. However, in this study, a demo version of the FAE, which only allows animating a simple face model, was used. The smooth-shaded version of this 3D model, which is called as “Mike”, is shown in the Figure 8. It is a very simple model composed of 408 vertices and 750 polygons.

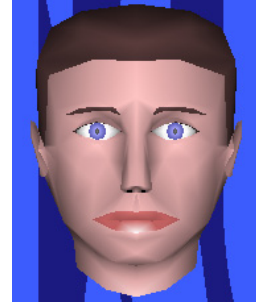


Figure 8 The smooth-shaded version of the 3D MPEG-4 face model “Mike” used for animation

5. Experiments and Evaluations

For our experiments, 10 minutes of audio-visual data from a female Turkish speaker was recorded. The corpus is a phonetically balanced Turkish speech corpus [21] containing 200 utterances. For the training of the system 150 utterances were used, and 50 utterances were set aside for testing purposes. This yielded 4.2 min of training data and 1.5 min of test data after removing the silence sections.

To evaluate the synthesis performance objectively, performance tests were made with different system parameters. The prediction performances of the synthesized facial parameters were determined by the correlation coefficient between the original and the synthesized facial trajectories. In addition, a performance measure PM was defined to measure the overall system performance.

$$PM = \frac{\sum_{k=1}^K e_k P_k}{\sum_{k=1}^K e_k} \quad (10)$$

In the above formulation, P_k is the performance metric for the k th PC, that is, the correlation between synthesized and predicted PC coefficient; e_k is the corresponding eigenvalue, and K is the total number of PCs used. This formulation provides a reasonable performance measure by weighting performance of synthesized PCs according to their variance. Tests were performed by varying the number of top most similar context-phones with different context-levels. Also, training utterances were synthesized and compared to see the theoretical upper limit of the performance. The results are shown in Figure 9. It is seen that, the effect

of the context-level is the same after three, but there is no significant difference after the context-level of one. This means, triphones are sufficient to handle coarticulation effects. Since the neighboring phonemes have the greatest influence on the coarticulation, this result is quite reasonable. According to the results, with the increasing number of top most similar context-phones, the performance is rising up to some point for the test set, while it is falling for the training set as can be expected. Of course, for a much larger training corpus, the prediction performance can be expected to increase with higher context lengths by capturing a large amount of coarticulation effect from the data. With the help of these tests, it was found that the optimal context-level is three, and the number of top most similar context-phones is six. Correlation values with these optimal parameters are listed in table 1.

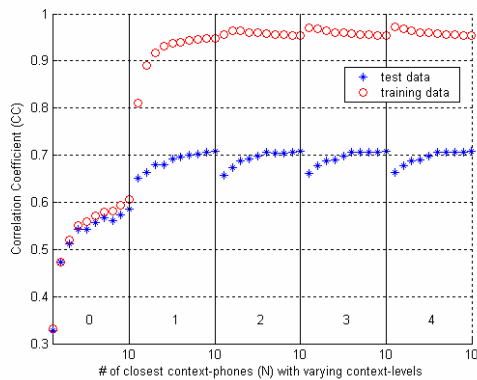


Figure 9 The effect of top most similar context-phones on the performance. Each column represents a different context-level.

Test Condition	CC
Training data (raw)	0.9578
Training data (smoothed)	0.9601
Test data (raw)	0.7055
Test data (smoothed)	0.7538

Table 1 Synthesis performance for context-level of 3 using top 6 visually most similar context-phones

Average CC	Raw		Smoothed	
	Test	Training	Test	Training
Lower lip (y)	0.6802	0.9289	0.7373	0.9533
Upper lip (y)	0.7600	0.9674	0.8097	0.9721
Whole lip (z)	0.7478	0.9614	0.7976	0.9589
Lip corners (x)	0.7074	0.9532	0.7620	0.9654

Table 2 Average correlation between synthetic and original face point trajectories for test and training data

In order to assess the synthesis of some important facial movements for speech articulation, average correlation values for corresponding face points are listed in table 2. These facial actions are lower and upper lip movements in y dimension that determine mouth opening; lip z displacement for the lip protrusion; and lip corner movements along x dimension crucial for lip rounding.

In Figure 10 an example synthesis result is shown by plotting the original, synthesized and smoothed trajectories. It belongs to test utterance “rol yapmam gerekmiyordu”, and the trajectories shows the center upper and lower lip vertical displacements with respect to the neutral positions. It is seen that the smoothed synthesized results are sufficiently good; trajectories pass through the proper locations, although the initial synthesized trajectories are noisy. For example, the phoneme /p/ is correctly synthesized by the closure of lower and upper lips; while the upper lip point moves downward, other one move upwards. On the other hand, over the silence region at the beginning of the utterance, a larger error is seen. However, this is due to the inhaling of the speaker, and can be ignored since it has no importance for the speech articulation.

We tried the system with different Turkish speakers. Although it was trained with audio-visual data from one female speaker and audio data from a small number of speakers, the synthesized animations seem realistic.

6. Conclusion

In this paper, a system that generates visual speech by synthesizing 3D face points using speech input was described. The synthesized face points were used to drive an MPEG-4 facial animation engine. To train the system with audio-visual data from a speaker, a 3D facial motion capture system was developed. Also, audio-only data from a small number of speakers were used for the training in order to improve the speaker-independent performance. According to objective tests, the system is quite successful in the prediction of articulatory speech movements. In addition, it can be used by any Turkish speaker to animate faces.

As future work, we are planning to add tongue information and perform subjective tests with the hearing-impaired for better evaluation of the system. Such a system can be used in a lip-reading tutorial for the speech therapy, or can be employed for user interface agents after integrating with text-to-speech.

7. References

- [1] McGurk, H., and MacDonald, J., “Hearing lips and seeing voices”, *Nature*, vol. 264, pp. 746-748, Dec. 1976.
- [2] Beskow, J., “Rule-based Visual Speech Synthesis”, *Proceedings of the 4th European Conference on Speech Communication and Technology (Eurospeech’95)*, Madrid, Spain, pp.299-302, 1995.

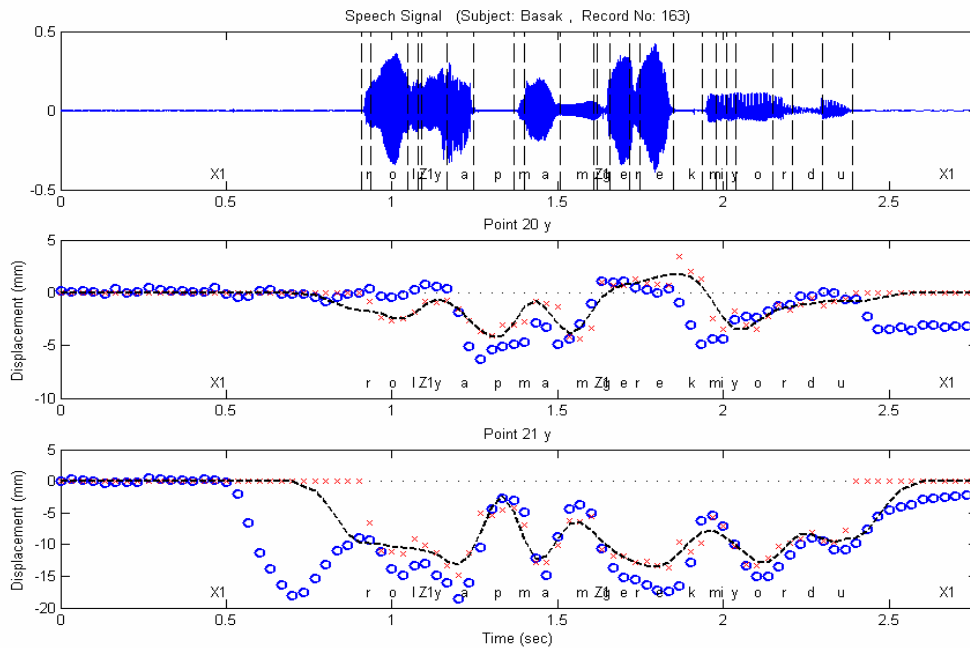


Figure 10 The comparison of original and synthesized face point displacement trajectories for test data. Top plot is speech signal of the utterance “rol yapmam gerekmiyordu”. Middle and bottom plots are the center upper lip and center lower lip y-axis trajectories respectively. Circle dotted curve: original; cross dotted curve: synthesized without smoothing; dashed curve: synthesized with smoothing.

- [3] Cohen, M. M., & Massaro, D. W., “Modeling coarticulation in synthetic visual speech”, in Thalmann, N. M., Thalmann, D. (Eds.), *Models and Techniques in Computer Animation*, Springer Verlag, Tokyo, pp. 139-156, 1993.
- [4] Bregler, C., Covell, M., Slaney, M., “Video rewrite: Visual speech synthesis from video”, *Proceedings of the Workshop on Audio-Visual Speech Processing*, Rhodes, Greece, pp. 153-156, 1997.
- [5] Lewis, J. P., and Parke, F. I., “Automatic lip-synch and speech synthesis for character animation”, *Proc. Graphics Interface '86*, pp.136-140, Canadian Information Processing Society, Calgary, 1986.
- [6] Yamamoto, E., Nakamura, S., and Shikano, K., “Lip movement synthesis from speech based on Hidden Markov Models”, *Journal of Speech Communication*, vol. 28, pp. 105-115, 1998.
- [7] Beskow, J., “Talking Heads: Models and Applications for Multimodal Speech Synthesis”, Ph.D. Thesis, Department of Speech, Music and Hearing, KTH. Stockholm, Sweden, 2003.
- [8] Arslan, L. M, and Talkin, D., “Codebook based face point trajectory synthesis algorithm using speech input”, *Elsevier Science*, 953, 01-13, December 1998.
- [9] Point Grey Research Inc., <http://www.ptrey.com/>, 2004.
- [10] Savran, A., “Speech and Text Driven 3D Face Synthesis for the Hearing-Impaired”, M.S. Thesis, Bogazici University, 2004.
- [11] Gonzalez, R. C., and Woods R. E., *Digital Image Processing*, Prentice Hall, 2002, ch 6, pp. 295-302.
- [12] Shapiro L. G., and Stockman, G. C., *Computer Vision*, Prentice Hall, 2001, ch 3, pp. 74-75.
- [13] Arun, K. S., Huang, S. T., and Blostein, S. D., “Least-squares fitting of two 3-D points sets”, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698-700, Sep. 1987.
- [14] Rabiner, L. R., and Schafer R. W., *Digital Processing of Speech Signals*, Prentice Hall, 1978.
- [15] Rothweiler, J., “On polynomial reduction in the computation of LSP frequencies”, *IEEE Transaction on Speech and Audio Processing*, vol. 7, no.5, pp. 592-594, Sep. 1999.
- [16] Rothweiler, J., “A root-finding algorithm for line spectral frequencies”, *Proceedings of the IEEE ICASSP 1999*, Phoenix, AZ, USA, II-661 – II-664, March 15-19, 1999
- [17] Huang, X., Acero, A., Hon, H. W., “*Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*”, Prentice Hall PTR, 2001, ch 6, pp. 316-318.
- [18] Unser, M., Aldroubi, A., and Eden, M., “B-Spline Signal Processing: Part I-Theory”, *IEEE Transaction on Signal Processing*, vol. 41, no. 2, pp. 821-833, Feb. 1993.
- [19] MPEG-4 Overview, ISO/IEC JTC1/SC29/WG11 N4668, March 2002.
- [20] Lavagetto, F., and Pockaj,R., “The FacialAnimation Engine: towards a high-level interface for the design of MPEG-4 compliant animated faces”, *IEEE Trans. Circuits and Systems for Video Technology*, vol. 9, no. 2, pp. 277-289, 1999.
- [21] Dutagaci, H., “Statistical Language Models for Large Vocabulary Turkish Speech Recognition”, M.S. Thesis, Bogazici University, 2002.