

Audio-Visual Unit Selection for the Synthesis of Photo-Realistic Talking-Heads

Eric Cosatto¹, Gerasimos Potamianos² and Hans Peter Graf¹

¹ AT&T Labs-Research, 100 Schulz Drive, Red Bank, NJ 07701, USA; {eric,hpg}@research.att.com

² presently at IBM, T.J. Watson Research Center, Yorktown Heights, NY 10598, USA; makis@watson.ibm.com

ABSTRACT

This paper investigates audio-visual unit selection for the synthesis of photo-realistic, speech-synchronized talking-head animations. These animations are synthesized from recorded video samples of a subject speaking in front of a camera, resulting in a photo-realistic appearance. The lip-synchronization is obtained by optimally selecting and concatenating variable-length video units of the mouth area. Synthesizing a new speech animation from these recorded units starts with audio speech and its phonetic annotation from a text-to-speech synthesizer. Then, optimal image units are selected from the recorded set using a Viterbi search through a graph of candidate image units. Costs are attached to the nodes and arcs of the graph that are computed from similarities in both the acoustic and visual domain. While acoustic similarities are computed by simple phonetic matching, visual similarities are estimated using a hierarchical metric that uses high-level features (position and sizes of facial parts) and low-level features (projection of the image pixels on principal components of the database). This method preserves coarticulation and temporal coherence, producing smooth, lip-synched animations. Once the database has been prepared, this system can produce animations from ascii text fully automatically.

Keywords: facial animation, talking-heads, sample-based image synthesis, computer vision.

1. INTRODUCTION

Most methods for the synthesis of animated talking heads use models that are parametrically animated from speech. Earlier methods derive parameter values from coarticulation modules such as [3]. Several head models have been demonstrated, including texture-mapped 3D models [7][10] and parameterized 2.5D models [4]. More recently, researchers have devised methods to learn parameters and their movements from labeled voice and video data [1]. Very smooth-looking animations have been produced by [5], using image morphing driven by pixel-flow analysis.

Another approach, inspired by recent developments in speech synthesis [12], is the so-called sample-based, image-driven or concatenative technique. The basic idea is to concatenate pieces of recorded data to produce new data. As simple as it sounds, there are many difficulties associated with this approach. The main difficulty is that a large, clean database is needed from which we can draw the samples. Bregler et al. reported a method for concatenative sample-based talking-head animations for dubbing video sequences [2]. Our approach shares some aspects of this work. Key differences are that our method does a full 3D modeling of the head and that we use a PCA-based metric for discriminating fine differences in the appearance of facial features. Bregler et al. use measurements of lip height and width as well as teeth visibility as visual features for unit selection. These features do not fully characterize the mouth. For example, pressure put on

the lips and presence of the tongue, or of the lower and upper teeth all influence the appearance of the mouth. The human visual system is extremely sensitive to minute variations in a face's appearance. Neglecting these fine details may lead to animations containing noticeable artifacts. We use a combination of geometric and pixel-based metrics to characterize the appearance of facial parts plus a full 3D head-pose estimation to compensate for different orientations. This enables our system to find similar-looking mouth images from the database, making it possible to synthesize smooth animations. In this way, we avoid having to morph dissimilar frames into each other, an operation that affects adversely the lip synchronization. Another key difference is that Bregler et al. use triphone segments as units of video. Instead of segmenting the video sequences a priori, we let the unit selection process find dynamically the best segments. This additional flexibility helps the synthesizer use longer contiguous segments of original video, resulting in animations that are more lively and pleasing.

Our system is divided into two parts: off-line processing to prepare the database and on-line processing for synthesis (Fig. 1).

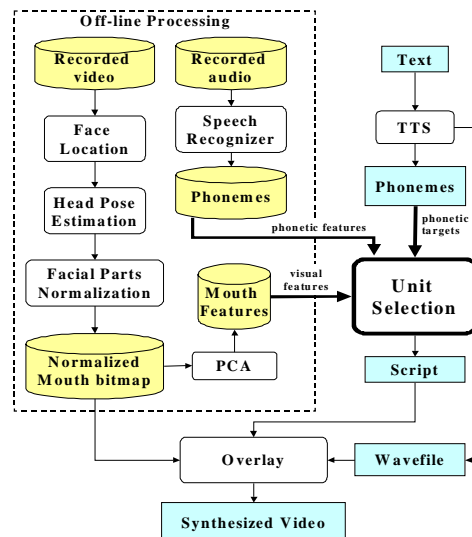


Fig. 1. Talking-head synthesis system overview.

The off-line part consists of processing the input video and audio into a format suitable for efficient unit selection. Normalized mouth bitmaps are extracted from the video frames, features are extracted from the normalized mouth bitmaps and the audio speech is phonetically labeled. Section 2 discusses the process of feature extraction and database setup. The on-line part starts with a request by the user in the form of ascii text from which the text-to-speech synthesizer (TTS) produces an audio track and a phonetic transcript. The system then performs unit selection to identify the most suitable segments of video from the

database and concatenates them into an animation (section 3). No recognition is involved in the on-line part making the system suitable to operate fully automatically in unsupervised environments. Results are discussed in section 4.

2. FEATURE EXTRACTION

Obtaining robust visual features from videos of a talking person is no simple task. Since parts of the recorded images are used to generate new images, the locations of facial features have to be determined with sub-pixel accuracy. Use of props or markers to ease feature recognition and tracking results in images that have to be post-processed to remove these artifacts, in turn reducing their quality. Part of the difficulty arises from letting subjects move their heads naturally while speaking. Early experiments with subjects whose heads were not allowed to move, resulted in animations that looked unnatural. On the other hand, letting the subject speak in front of the camera with neither head restraints nor any facial markers forces us to use advanced computer vision techniques to recognize and factor out the head pose before extracting features with high accuracy. Using the head pose, a normalized view of the area around the mouth can be obtained before applying a second round of feature extraction. Hierarchical feature extraction allows using low-level features that require image registration.

2.1 Normalization

The first step in obtaining normalized mouth bitmaps is to locate the face on the recorded videos. A wide variety of techniques exist to perform this task. We use the model-based, multi-modal, bottom-up approach reported in [6]. Shape, color and motion channels are used to estimate the position of facial features such as eyes, nostrils, mouth, eyebrows and head contour. Candidates for these parts are found from connected pixels and are scored using n-grams against a standard model. The highest scoring combination is taken to be a head, giving the positions of eyes and nostrils on the image. A second pass uses specialized, learned convolution kernels to obtain a more precise estimate of the position of sub-parts such as eye-corners.

To find the position and orientation of the head, we apply the pose estimation technique reported in [9]. We first obtain a rough 3D model of the subject containing at least 4 coplanar points. For added precision, we use 6 points, namely: the four eye corners and the two nostrils. These points are measured manually on calibrated photographs of the subject's face (frontal and profile views). Next, we get the corresponding positions of these points in the image from the face recognition module. Pose estimation starts with the assumption that all model points lie in a plane parallel to the image plane (corresponds to a orthographic projection of the model into the image plane plus a scaling). Then, by iteration, the algorithm adjusts the model points until their projections into the image plane coincide with the observed image points. The pose of the 3D head model (referred to as 'object' in the following) is obtained by solving iteratively the following linear system of equations:

$$\begin{cases} \mathbf{M}_k \cdot \frac{f}{Z_0} \mathbf{i} = x_k(I + \varepsilon_k) - x_0 \\ \mathbf{M}_k \cdot \frac{f}{Z_0} \mathbf{j} = y_k(I + \varepsilon_k) - y_0 \end{cases}$$

\mathbf{M}_k is the 3D position of object point k , \mathbf{i} and \mathbf{j} are the two first base vectors of the camera coordinate system in object coordinates, f is the focal length and Z_0 is the distance of the object origin from the camera. \mathbf{i} , \mathbf{j} , and Z_0 are the unknown

quantities to be determined. (x_k, y_k) is the scaled orthographic projection of the model point k , (x_0, y_0) is the origin of the model in the same plane, ε_k is a correction term due to the depth of the model point. ε_k is adjusted at each iteration until the algorithm converges.

This algorithm is numerically very stable, even with measurement errors, and it converges in just a few iterations. Using the recovered angles and position of the head, we project a 3D plane bounding the facial part onto the image plane. The resulting quadrilateral is used to warp the bounded pixels into a normalized bitmap. This operation is done for each facial part needed for the synthesis. In the rest of the text, we will focus on the mouth area.

2.2 Principal Components

A set of features is extracted from the normalized mouth images that can be used to distinguish them efficiently. We are interested in features that define a space in which the Euclidian distance between two images can be directly related to their difference as perceived by a human observer. Ultimately our goal is to find a metric that enables the unit selection module to generate *smooth* talking-head animations by selecting frames from the database that are *visually close* (Fig. 2).

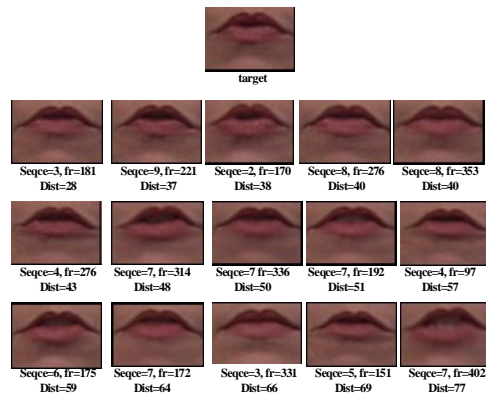


Fig. 2. Principal components as a distance metric (Euclidian distance in the space of principal components). 15 closest mouths to the given target.

Wavelet representation, Principal Component Analysis and Linear Discriminant Analysis have been used successfully for lip-reading [11] and have been demonstrated to capture reliably the appearance of mouth and lip images. Here we choose Principal component analysis (PCA) since it provides a compact representation and captures the appearance of the mouth with just a few parameters. Images are first sub-sampled and separated into 3 channels (red, green, blue of which typically only the red channel is used). Each resulting image is packed into a vector and vectors are stacked into a data matrix. If the size of an image vector is n and the number of images is m , then the data matrix M is an nxm matrix. PCA is performed by calculating the eigenvectors of the $n \times n$ covariance matrix of M . The process of feature extraction is now reduced to projecting a vector onto the first few principal components (eigenvectors with the largest eigenvalues). In our experiments we found that the first 12 eigenvectors provide sufficient discrimination to give a useful metric.

PCA is generally an expensive algorithm, especially for images, because of the large matrices that have to be computed.

Fortunately, because we can sub-sample the images, and because we perform a separate analysis for each color channel, the resulting size of the covariance matrix is kept within reasonable limits. In our experiments we sub-sampled our images down to 30 by 18 pixels, resulting in 3 vectors of size 540 per image. Our database contains over 20,000 images resulting in an input data matrix of around 50MB. The covariance matrix, however, depends only on the size of an image and is only slightly larger than 1MB. The complexity of obtaining the eigenvectors is of order $O(n^3)$. Since both the complexity and the memory requirements scale directly (and only) with the size of the image, it is advantageous to sub-sample the images.

2.3 Database

The original raw videos of our subjects articulating sentences are processed to extract the following files:

- video files of the normalized mouth area.
- some whole-head videos to provide background images.
- feature files for each mouth.
- phonetic transcripts of all sentences.

The size of the database is directly related to the quality required for the animations. Better lip-synch requires more sequences and higher image resolution makes larger files.

3. UNIT SELECTION

To synthesize a new animation, the input ascii text is first run through a text-to-speech synthesizer (TTS) generating the audio track and its phonetic transcript [12]. A video frame rate is chosen which, together with the length of the audio, determines the number of video frames that have to be synthesized. Each video frame is built by overlaying bitmaps of face parts to form a whole face using a method similar to the one in [4].

To achieve synchronization of the mouth with the audio track while keeping the resulting animation smooth and pleasing to the eye, we borrow a technique from concatenative speech synthesis: unit selection [8]. Unit selection is driven by two cost functions: a target cost and a concatenation cost. A database contains a set of units, which are labeled with features that can be used to compute the cost functions.

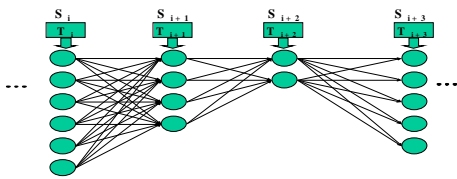


Fig. 3. Graph of an animation with states S_i , targets T_i and a list of candidates (ovals) for each state. Each state generates a target cost, while each transition (arc of the graph) generates a concatenation cost.

A graph with n states corresponding to the n frames of the final animation is built (Fig. 3). Each state contains a list of candidate images from the database and is fully connected to the next state. Each candidate has a *target cost* and two consecutive candidates generate a *concatenation cost*. The number of candidates at each state is limited by a maximum target cost. A Viterbi search through the graph finds the optimum path (the one that generates the lowest cost).

The task of the unit selection is to balance two competing goals. On the one hand we want to insure lip synchronization. Working towards this goal, the *target cost* uses phonetic and

visemic context to select a list of candidates that most closely match the phonetic and visemic context of the target. The context spans several frames to ensure that coarticulation effects are taken into account.

On the other hand we want to ensure smoothness in the final animation. To achieve this goal, we want to be able to use the longest possible original segments from the database. The *concatenation cost* works towards that goal by penalizing segment transitions and insuring that when we need to transition to another segment, we choose a candidate that is visually close to its predecessor, thus generating the smoothest possible transition. This cost has two distinct components (the skip cost and the transition cost) because the visual distance between two frames cannot be perfectly characterized (otherwise, the transition cost alone would suffice). The feature vector of an image provides only a limited, compressed view of its original, so that the distance measured between two candidates in the feature space cannot always be trusted to ensure perfect smoothness of the final animation. The additional skip cost is a hint given to the system that consecutively recorded frames *are* smoothly transitioning.

3.1 Target Cost using Acoustic Features

The target cost is a measure of how much distortion a given candidate's features have compared to the target features. The target feature vector is obtained from the phonetic annotation of a given frame of the final animation. The target feature vector at frame t , $\mathbf{T}(t) = \{ph_{t-nl}, ph_{t-nl-1}, \dots, ph_{t-1}, ph_t, ph_{t+1}, \dots, ph_{t+nr-1}, ph_{t+nr}\}$ is of size $nl+nr+1$, where nl and nr are respectively the extent in frames of the coarticulation left and right of the target ph_t (the phoneme being spoken at frame t). A weight vector of the same size is given by $\mathbf{W}(t) = \{w_{t-nl}, w_{t-nl-1}, \dots, w_{t-1}, w_t, w_{t+1}, \dots, w_{t+nr-1}, w_{t+nr}\}$,

$$w_i = e^{-\alpha|t-i|}, i \in [t-nl; t+nr]$$

This weight vector simulates coarticulation by giving an exponentially decaying influence to phonemes, as they are further away from the target phoneme. The values of nl , nr and α are not the same for every phoneme. A table look-up is used to get the particular values for each target phoneme. An interesting case is the silence phoneme. Coarticulation might extend much more during a silence preceding speech than during speech itself, requiring nl and nr to be larger and α smaller. This simplified coarticulation model has been inspired by a more elaborate model published in [3].

For a given target and weight vector, the whole database is now searched to find the best candidates. A candidate extracted from the database at frame u has a feature vector $\mathbf{U}(u) = \{ph_{u-nl}, ph_{u-nl-1}, \dots, ph_{u-1}, ph_u, ph_{u+1}, \dots, ph_{u+nr-1}, ph_{u+nr}\}$. It is compared with the target feature vector. The target cost for frame t and candidate u is given by:

$$TC(t, u) = \frac{1}{\sum_{i=-nl}^{nr} w_{t+i}} \sum_{i=-nl}^{nr} w_{t+i} \cdot M(T_{t+i}, U_{u+i})$$

where $M(ph_i, ph_j)$ is a $p \times p$ "viseme distance matrix" where p is the number of phonemes in the alphabet. This matrix denotes visual similarities between phonemes. For example the phonemes {m,b,p}, while different in the acoustic domain have a very similar appearance in the visual domain and their "viseme distance" will be small. This matrix is populated with values inspired by the literature on visemes. Hence the target cost TC measures the distance of the *audio-visual coarticulation context* of a candidate to that of the target. To reduce the complexity of

the Viterbi search, we set a maximum number of candidates at each state. Candidates with the smallest target cost are considered.

3.2 Concatenation Cost using Visual Features

Once candidates have been selected for each state, the graph of Fig. 3 is constructed and each arc is given a concatenation cost that measures the distance between a candidate of a given state and a candidate of the previous state. Both candidates $u1$ (from state i) and $u2$ (from state $i-1$), have a feature vector $U1$, $U2$, calculated from the projection of their respective image (pixels) onto the k first principal components of the database (see section 2). The concatenation cost is given by $CC(u1,u2)=f(U1,U2) + g(u1,u2)$, where:

$$f(U1,U2) = \frac{1}{\sqrt{k}} \sqrt{\sum_{i=1}^k (U1_i - U2_i)^2}$$

is the Euclidian distance in the feature space. This cost reflects the visual difference between two candidate images as captured by the chosen features. The other cost:

$$g(u1,u2) = \begin{cases} 0 & \text{when } fr(u1) - fr(u2) = 1 \wedge seq(u1) = seq(u2) \\ w_1 & \text{when } fr(u1) - fr(u2) = 0 \wedge seq(u1) = seq(u2) \\ w_2 & \text{when } fr(u1) - fr(u2) = 2 \wedge seq(u1) = seq(u2) \\ \dots & \\ w_{p-1} & \text{when } fr(u1) - fr(u2) = p-1 \wedge seq(u1) = seq(u2) \\ w_p & \text{when } fr(u1) - fr(u2) \geq p \vee fr(u1) - fr(u2) < 0 \\ & \vee seq(u1) \neq seq(u2) \end{cases}$$

where $0 < w_1 < w_2 < \dots < w_p$, $seq(u) = recorded_sequence_nb$ and $fr(u) = recorded_frame_nb$, is a cost for skipping consecutive frames of a sequence. This cost helps the system to avoid switching too often between recorded segments, thus keeping as much as possible of the integrity of the original recordings. We set $p=5$ and let the w_i increase exponentially. In this way, the small cost of w_1 and w_2 allows for varying the length of a segment by occasionally skipping a frame or repeating a frame to adapt its length (scaling). The high cost of w_5 , however, ensures that skipping more than 5 frames incurs a high cost, avoiding jerkiness in the resulting animation.

3.3 Viterbi Search

The graph $G=\{S_0, S_1, \dots, S_n\}$ of Fig. 3 with states S_i and candidates $S_{i,j}$ has been constructed with a target cost TC for each candidate and concatenative cost CC for each arc going from state S_i to state S_{i+1} . A path $\{p_0, p_1, \dots, p_n\}$ through this graph generates the following cost:

$$c = WTC \cdot \sum_{t=0}^n TC(t, S_{t,p_t}) + WCC \cdot \sum_{t=1}^n CC(S_{t,p_t}, S_{t-1,p_{t-1}})$$

The best path through the graph is the path that produces the minimum cost. The weights WTC and WCC are used to fine-tune the emphasis given to concatenation cost versus target cost, or in other words to emphasize acoustic versus visual matching. A strong weight given to concatenation cost will generate very smooth animation but the synchronization with the speech might be lost. A strong weight given to target cost will generate an animation which is perfectly synchronized to the speech but that might appear visually choppy or jerky due to the high number of jumping within database sequences. We are currently finding these coefficients by maximizing the average segment length over a set of training sequences.

4. RESULTS AND DISCUSSION

We produced talking-head animations that can be mistaken easily for real videos. Examples can be found on our web site at: <http://www.research.att.com/~eric/synthesis.html>

Many of the algorithms used in this system are borrowed from concatenative speech synthesis. In fact the TTS system we use is also concatenative [12], giving a natural voice to a naturally looking talking-head.

Of great importance for the visual quality of the animation is the size of the database and in particular, how well it targets the desired output. Best animations are produced when few fairly large segments (larger than 400ms) can be taken as a whole from the database within a sentence. For this to happen, the database must contain enough examples of speech. As an illustration, we have made experiments with numbers (0 to 9). The database for this task contains 4.16 minutes of speech (15000 frames recorded at 60fr/sec). From this database we are able to synthesize arbitrary sequences of numbers (for example telephone numbers) with a quality deemed "natural" or "very natural" by over 70% of viewers. In this case the average segment length is 400ms. On the other hand, general text synthesized from this database has an average segment length of 170ms and a poorer visual quality. For general text, we have experimented with a small database (2 minutes of phonetically balanced sentences). While the results may be sufficient for some applications, for highest quality we need a larger database.

References

- [1] Brand, M., **Voice Puppetry**, ACM SIGGRAPH, 1999.
- [2] Bregler, C., Covell, M., Slaney, M., **Video Rewrite: Driving Visual Speech with Audio**, ACM SIGGRAPH, 1997.
- [3] Cohen, M.M., Massaro, D.W., **Modeling Coarticulation in Synthetic Visual Speech**, Models and Techniques in Computer Animation, Springer Verlag, 1993.
- [4] Cosatto, E., Graf, H.P., **Sample-Based Synthesis of Photo-Realistic Talking-Heads**, IEEE Computer Animation, 1998.
- [5] Ezzat, T., Poggio, T., **MikeTalk: A Talking Facial Display Based On Morphing Visemes**, IEEE Computer Animation, 1998.
- [6] Graf, H.P., Cosatto, E., Potamianos, G., **Robust Recognition of Faces and Facial Features with a Multi-Modal System**, IEEE Systems, Man and Cybernetics, pp. 2034-2039, 1997.
- [7] Guenter, B., Grimm, C., Wood, D., Malvar, H., Pighin, F., **Making Faces**, ACM SIGGRAPH, pp. 55-66, 1998.
- [8] Hunt, A., Black, A., **Unit selection in a concatenative speech synthesis system using a large speech database**, ICASSP, vol. 1, pp. 373-376, 1996.
- [9] Oberkampff, D., Dementhon, D., Davis, L., **Iterative Pose Estimation Using Coplanar Feature Points**; Internal Report, CVL, CAR-TR-677, University of Maryland, 1993.
- [10] Ostermann, J. **Animation of synthetic faces in MPEG-4**, IEEE Computer Animation, pp. 49-55, 1998.
- [11] Potamianos, G., Graf, H.P., **Linear Discriminant Analysis for Speechreading**, IEEE Workshop on Multimedia Signal Processing, pp. 221-226, 1998.
- [12] Beutnagel, M., Conkie, A., Schroeter, J., Stylianou, Y., Syrdal, A., **The AT&T Next-Gen TTS System** J. Acoust. Soc. Am., Vol. 105, Pt. 2, p. 1030, 1999.