Multimodal Speaker Conversion — his master's voice…and face —

T. Dutoit*, A. Holzapfel†, M. Jottrand*, F. Marqués‡, A. Moinet* F. Ofli§, J. Pérez‡ and Y, Stylianou†
*Faculte Polytechnique de Mons - BELGIUM †University of Crete - GREECE ‡Universitat Politècnica de Catalunya - SPAIN §Koc University - TURKEY

Abstract— **The goal of this project is to convert a given speaker's speech (the Source speaker) into another identified voice (the Target speaker) as well as analysing the face animation of the source to animate a 3D avatar imitating the source facial movements. We assume we have at our disposal a large amount of speech samples from the source and target voices with a reasonable amount of parallel data. Speech and video are processed separately and recombined at the end.**

Voice conversion is obtained in two steps: a voice mapping step followed by a speech synthesis step. In the speech synthesis step, we specifically propose to select speech frames directly from the large target speech corpus, in a way that recall the unit-selection principle used in state-of-the-art text-to-speech systems.

The output of this four weeks work can be summarized as: a tailored source database, a set of open-source MATLAB and C files and finally audio and video files obtained by our conversion method. Experimental results show that we cannot aim to reach the target with our LPC synthesis method; further work is required to enhance the quality of the speech.

Index Terms— **voice conversion, speech-to-speech conversion, speaker mapping, face tracking, cloning, morphing, avatar control.**

I. Introduction

THIS project aims at converting a given speaker speech and facial movements into those of another (identified) speaker. More precisely, it focuses on controlling a 3D talking face (the avatar of the target speaker) using the most natural interfaces available : the speech and facial movements of a human speaker (the source speaker). It also assumes that the target will generally be different from the source, so that the project goes much further than the design of a state-of-the-art avatar controlled by the original speaker (i.e., the same as the one whose face was used to create the avatar). As a matter of fact, two additional problems are encountered here:

- That of voice conversion, in order to make the target talking face speak with the target's voice, producing the source's words.
- That of facial movement conversion, in order to adapt the movement excursions of the source face to match the movement excursions of the target face.

The multimodal conversion problem we consider here, however, is limited to signal-level modification, as opposed to semantic conversion. The latter would enable, for instance, filtering out some (most often paralinguistic) communication acts (be them speech and/or facial movements, such as tics) of the source which the target never produces, or conversely adding target movements not present in the source but usually in the target.

Moreover, we constraint the conversion process to maintain some large-sense synchronicity between source and target: we do not aim at adapting speech rate at the phoneme level, but rather simplifying it to a possible overall speech rate adaptation. Similarly, we do not consider a possible syllable-level F0 conversion from source to target, but rather aim at a possible overall F0 adaptation ratio.

It will be assumed that a large amount of studio-quality speech data is available from the source *and* from the target. This is not a usual assumption for systems which try to put new words in the (virtual) mouth of most VIP characters (whom cannot be easily forced to attend a formal recording session in a studio). The assumption, however, remains realistic in the case of a source speaker driving a famous target speaker whose voice has been recorded in large amounts but who is simply no longer (or not always) available. It is also assumed that this speech data is available in the form of parallel speech corpora (i.e., recordings of the same sentences by both the source and the target).

A typical application of this project is therefore that of a human actor controlling the speech and facial movements of a 3D character whose voice is well-known to the audience. Another possible use is for psychologists talking to children through an avatar whose voice should be kept unchanged among sessions, even though the psychologist may change. If we reduce this project to its speech component, a typical application is that of a tool for producing natural sounding voice prompts with the voice of a voice talent, based on natural speech (prosody and spectral features) produced by a human speaker.

Last but not least, a side constraint of this work is that we aim at using and producing open-source code, as required by the eNTERFACE workshop organization.

Figure 1 shows the necessary steps involved in our project: speech/face analysis, voice/facial movements mapping, and speech/face synthesis. This report is therefore organized as follows. Section II browses the state-of-the-art in speech analysis, mapping, and synthesis for voice conversion, and examines the approach we have followed in this project. Section III examines facial movement analysis, mapping and synthesis for avatar face animation, and gives details on the algorithms we have used. This is followed in section IV by

This report, as well as the source code for the software developed during the project, is available online from the eNTERFACE'06 web site: http://www.enterface.net/enterface06.

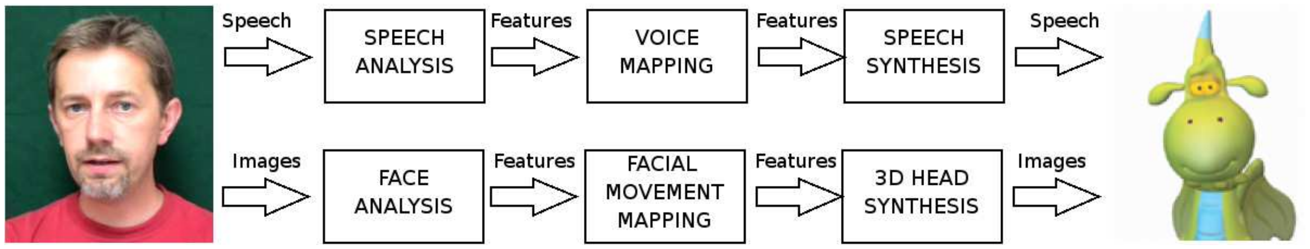

Fig. 1. An example of 3D cartoon avatar control using source speech and facial movements, and mapping them to target speech and face.

the experiments we did, using a specially designed database, and by some informal assesment of the quality we reached at the end of the 4-weeks project. The paper is concluded in Section V by perspectives for further developments.

II. Speech analysis, mapping, and synthesis for voice conversion

We understand the Voice Conversion (VC) process as a modification of a speaker voice (*source speaker*) so that it resembles that of another given speaker (*target speaker*). Our goal is to obtain a transformation that makes the source speech sound as if it were spoken by the target speaker.

In this work, we have approached the speech conversion task from x (source speaker) to y (target speaker) in two independent blocks:

- mapping from x to y' (a first approximate of y) using the (reduced) parallel corpus, and
- speech-to-speech (S2S) synthesis from y' to y'' (a second –and more accurate– approximate of y), using the (full) target corpus.

The first block involves aligning the data on a frame by frame basis (section II-A) and building a mapping function (either using Gaussian Mixture Models or Conditional Vector Quantization, as will be seen in section II-B). In the second block, for the successive frames of y', we select new frames from a large database of y voice, in such a way that we guarantee both maximum similarity to the input frames, and maximum continuity (the details are given in section II-D.3). This can be seen as a smoothing step, made necessary by the fact that the reduced size of the parallel corpus resulted in large, non-acceptable discontinuities in the synthetic speech.

It is worth mentioning that since a large amount of data for the target is available for this particular application (as it is also the case for the design of a state-of-the-art text-to-speech (TTS) system), the challenge of this project is to be able to produce more natural sounding speech than that of a TTS system, trained with the same amount of data, and used to synthesize the phonemes obtained by automatic phonetic segmentation (speech recognition) of the input source waveform. As a matter of fact, the input of the voice conversion system is itself natural speech, which can hopefully be put to profit to deliver natural-sounding output speech. The intonation of the source speech, for instance, can readily be used (possibly modified) to produce the intonation of the target speech, and thus obtain an improvement in synthesis quality over standard TTS.

Furthermore, it is then possible to establish upper and lower bounds to the synthesis quality we can obtain with our frame-based voice conversion system. The lower bound would be that obtained with a TTS backend. In this case, we would directly use the prosody of the source waveform and the phonemes, obtained with automatic speech recognition. The TTS would then only need to perform unit selection and concatenation based on this requirements. This way we expect to obtain a higher similarity and naturality than when using only the source (recognized) text as input. The upper bound would of course be the natural target speech.

A. Data alignment

Although the corpus used for the voice mapping part of +this project (see Section IV) consists of parallel utterances, some timing differences are unavoidable due to different speaker characteristics (pause durations, speech rate, etc.). Since the training of the voice mapping block of fig. 1 requires parallel data vectors, the utterances of the source and target speakers have been aligned using a dynamic time warping (DTW) procedure. For this project, we used the DTW algorithm implemented by Dan Ellis[1] and released under GPL. The *local match* measure is computed as the cosine distance (angle of the vectors) between the Short-Time Fourier Transform (STFT) magnitudes. The left part of fig. 2 represents the local match scores matrix, with darker zones indicating high similarity values (ideally we would have a dark stripe down the leading diagonal).In the right part of the figure we can see the minimum-cost-to-this-point matrix (lighter color indicates lower cost). In both subfigures, the red line shows the lowest-cost path obtained by the DTW algorithm.

The DTW algorithm introduces some unavoidable errors due to the coexistence of intrinsic spectral differences between the two speakers. An iterative procedure can be applied to improve the alignment, by reapplying the DTW method between the converted and target envelopes [1]. After each iteration, a new mapping function can be estimated between the newly aligned original source and target data. In this project, convergence was reached after three iterations.

B. Voice mapping

When converting the voice of the source to the voice of a the target speaker we assume that these two voices are defined by their spectral spaces $\mathcal{X}$ and $\mathcal{Y}$ respectively. Our problem

[1]http://labrosa.ee.columbia.edu/matlab/dtw/

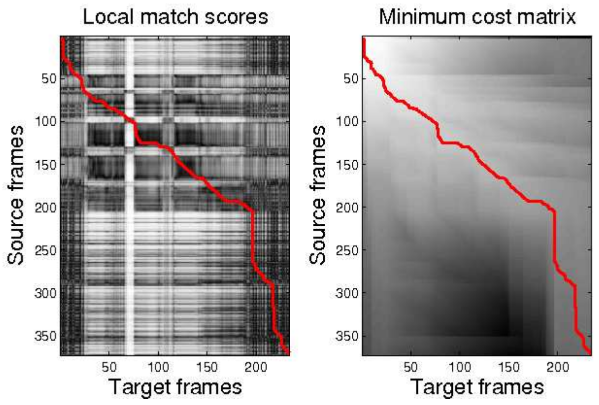

Fig. 2. Left: Graphical representation of the local match scores matrix (darker colors represent a closer match). Right: Minimum cost matrix (lighter colors indicate a lower cost). The red line shows the optimal alignment path as found by the DTW algorithm.

in voice conversion is two-fold: at first we have to find a way to model these spaces and then we have to find a way to map a previously unknown example from the source space to the target space. In order to be able to find such a mapping we assume that there is aligned training data available. This means that we have two sets of spectral vectors $\mathbf{x}_t$ and $\mathbf{y}_t$ that describe spectral envelopes from source and target speakers respectively. The two sets of vectors $\{\mathbf{x}_t,\ t = 1, \ldots, N\}$ and $\{\mathbf{y}_t,\ t = 1, \ldots, N\}$ have the same length N and are supposed to describe sentences uttered in parallel by source and target. What is desired is a function $\mathcal{F}()$ such that the transformed envelope $\mathcal{F}(\mathbf{x}_t)$ best matches the target envelope $\mathbf{y}_t$, for all envelopes in the learning set ($t = 1, \ldots, N$).

In our project we tried two different approaches in order to achieve the goal of conversion: Gaussian Mixture Models and Conditional Vector Quantization.

a) Gaussian Mixture Models: The first approach has been described by Stylianou *et al.*[1] and is based on a description of the source space using Gaussian Mixture Models:

$$p(\mathbf{x}) = \sum_{i=1}^{M} \alpha_i \mathcal{N}(\mathbf{x}; \mu_i, \mathbf{\Sigma}_i) \tag{1}$$

where M is the number of Gaussians, $\mu_i, \mathbf{\Sigma}_i$ are the mean vector and the covariance matrix of the i-th Gaussian component, and α_i are the weights used to combine the different components. These M Gaussian components can be regarded as classes within the spectral space of the source and a vector $\mathbf{x}_t$ can be classified to one of the classes using maximum likelihood. The mapping to the target space is done by using these parameters in the conversion function

$$\mathcal{F}(\mathbf{x}_t) = \sum_{i=1}^{M} P(\mathcal{C}_i|\mathbf{x}_t) \left[\nu_i + \mathbf{\Gamma}_i \mathbf{\Sigma}_i^{-1}(\mathbf{x}_t - \mu_i)\right] \tag{2}$$

where ν and $\mathbf{\Gamma}$ are related to the mean target and the cross-covariance matrix of the source and target vectors. The parameters of the conversion function are determined by minimization of the total quadratic spectral distortion between the converted envelopes and the target envelopes:

$$\epsilon = \sum_{t=1}^{N} ||\mathbf{y}_t - \mathcal{F}(\mathbf{x}_t)||^2. \tag{3}$$

For details on the minimization see [1].

b) Conditional Vector Quantization: The second method for voice conversion applies a Conditional Vector Quantization as presented in [2]. In contrast to the first method we get hard cluster boundaries by using a standard LBG clustering for the source space giving us a codebook $C_x \equiv \{\dot{\mathbf{x}}_i\ , i = 1...m\}$. Then the mapping function finds for each of these clusters a different codebook C_y with k entries for each source space cluster. The criterion function minimized in this case is the approximation to the average distortion D given by

$$D \approx \sum_{m=1}^{M} \left[\frac{1}{N} \sum_{n=1}^{N} p(\dot{\mathbf{x}}_m|\mathbf{x}_n) \sum_{k=1}^{K} p(\dot{\mathbf{y}}_{m,k}|\dot{\mathbf{x}}_m, \mathbf{y}_n) d(\mathbf{y}_n, \dot{\mathbf{y}}_{m,k})\right] \tag{4}$$

The conditional probability $p(\dot{\mathbf{x}}_m|\mathbf{x}_n)$ is the association probability relating the input vector $\mathbf{x}_n$ with codevector $\dot{\mathbf{x}}_m$, while the association probability $p(\dot{\mathbf{y}}_{m,k}|\dot{\mathbf{x}}_m, \mathbf{y}_n)$ relates the output vector $\mathbf{y}_n$ with the codevector $\dot{\mathbf{y}}_{m,k}$ of the m-th subcodebook of C_y.

The mapping for a source feature vector $\mathbf{x}_i$ is done by choosing the nearest source space cluster using Euclidean distance. This provides us with a subcodebook of C_y with K entries. For an utterance of length N we construct a lattice of $K \times N$ elements and we find a minimum weight path using again Euclidean distance from frame to frame providing us with a sequence of N vectors in $\mathcal{Y}$.

C. Frame selection algorithm

Once the features of the frames of the original speaker have been converted using either the GMM mapping or the CVQ-based conversion, the converted features are used as inputs of the unit-selection algorithm.

This algorithm is basically working like any TTS unit-selection system. However, state-of-the-art TTS systems based on unit-selection usually deal with diphones, phones or parts of phones [3] while our algorithm uses smaller units : 32 ms frames (with a constant shift of 8 ms between each frame).

For this part of the system, we use the complete target speech database (as opposed to the mapping system, which only used the aligned sub-part of it; see Section IV for details on the databases). Among all the frames in the target database (cmu_us_awb_arctic ’95), we select a sequence of frames $\ddot{Y} = [\ddot{y}^{(1)} ... \ \ddot{y}^{(t)} ... \ \ddot{y}^{(T)}]$ that best matches the sequence of frames output by the mapping function: $\dot{Y} = [\dot{y}^{(1)} ... \ \dot{y}^{(t)} ... \ \dot{y}^{(T)}]$. This selection is made using the Viterbi algorithm [4], [5] to minimize the global distance between $\dot{Y}$ and $\ddot{Y}$. This global distance is a combination of target and concatenation distances which are detailed in the next three subsections. This approach is very similar to that developed in Suenderman *et al.* [6], with the difference that in our approach the target sequence for the frame selection algorithm is the mapped sequence $\dot{Y}$, while Suenderman *et al.* use the input sequence X as target.”

1) Clustering, clusters and n-best selection: In order to reduce the computation time, the database has been divided into 256 clusters using the LBG method [7]. When we use only the parallel recordings as a database, there are about 300 frames per cluster and each cluster has a centroid which is a vector of the mean values of the features of the frames inside the cluster. Therefore, for each set of features $\dot{y}^{(t)}$, the algorithm first selects the cluster with the closest centroid. The closeness of the centroid is measured using a weighted euclidean distance

$$closest\ centroid = \underset{c=1,\dots,C}{\operatorname{argmin}} \sum_{i=1}^{N} w_i \cdot \left(\dot{y}_i^{(t)} - \ddot{y}_i^{(c)}\right)^2, \quad (5)$$

where N is the dimension of the feature vectors, $\dot{y}_i^{(t)}$ is the i^{th} component of the feature vector produced by the mapping function at time t, $\ddot{y}_i^{(c)}$ is the i^{th} component of the c^{th} centroid of the database and w_i is the weighting factor associated to that i^{th} component.

Then, for each frame $(1, ..., m_c, ..., M_c)$ in the chosen cluster c, the weighted euclidean distance between the feature vectors $\dot{y}^{(t)}$ and $\ddot{y}^{(m_c)}$ is computed. This distance will be used as the target distance t_{dist} by the Viterbi algorithm:

$$t_{dist}(t, m_c) = \sum_{i=1}^{N} w_i \cdot \left(\dot{y}_i^{(t)} - \ddot{y}_i^{(m_c)}\right)^2 \quad (6)$$

Finally, if the n-best option is activated, only the NBEST closest $\ddot{y}^{(m_c)}$ to the $\dot{y}^{(t)}$ are selected as candidates for Viterbi.

2) Concatenation distance: During the induction step, the Viterbi algorithm has to compute the cost of all the transitions from each feature vector $\ddot{y}^{(m_{c(t)})}$, selected at time instant t, to each vector $\ddot{y}^{(m_{c(t+1)})}$, selected at time $t + 1$. Again, these concatenation costs (c_{dist}) are measured using a weighted euclidean distance:

$$c_{dist}(m_{c(t)}, m_{c(t+1)}) = \sum_{i=1}^{N} w_i \cdot \left(\ddot{y}_i^{(m_{c(t+1)})} - \ddot{y}_i^{(m_{c(t)})}\right)^2 \quad (7)$$

This step is the most time-consuming part of the Viterbi algorithm and is the reason why clusters and n-best selections are so important. They reduce the transition possibilities but in return they dramatically increase the search speed without actually nor notably damaging the final output.

At this point, one important remark has to be done : the concatenation distance should advantage the selection of neighbour frames to reduce discontinuities during the synthesis of speech. Therefore, before computing the distance between two feature vectors, the process checks whether the corresponding frames are consecutive in a wav file of the database. In case they are, the distance is automatically set to zero.

3) Implementation: The very first tests were made using Matlab, however it early appeared that this was very time-consuming. In order to reduce the duration of the Viterbi algorithm, the whole program has been rewritten in C so that it could be compiled in a mex-file. Mex-files are pre-compiled libraries used by Matlab as any other Matlab script. The difference with usual scripts is that the functions implemented in these libraries are way faster (see Annex V-G in p. 11).

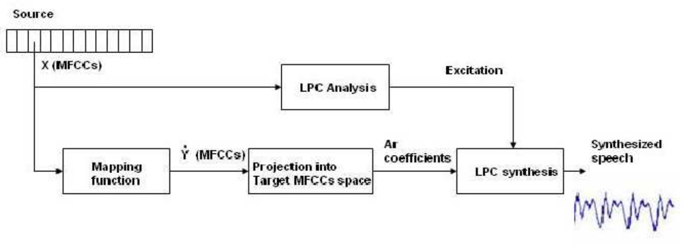

Fig. 3. Block diagram of the selection method using a projection of the mapped source MFCCs vectors into the target MFCCs space.

D. Speech synthesis

As a result of the previous steps (mapping, possibly followed by a frame-selection algorithm as described in II-C), we obtained a sequence of converted (target) frames that best represent the voice transformation of a given source utterance. The next step is to generate the converted speech resulting from this sequence. Although it is possible to regenerate the speech from the MFCC parameters [8], the resulting quality is not acceptable for the task, since the parametrization is not a lossless procedure. In particular, the phase information is completely lost and must be estimated to provide the synthetic voice with a higher degree of naturalness. In this subsection, we first produce $\ddot{y}(n)$ using the source speaker excitation as input of LPC filters modelling the target speech. We then examine the production of $\ddot{y}(n)$ by overlap-adding speech frames (i.e., samples) extracted directly from the target database.

1) LP-based speech synthesis, without Viterbi: This method is illustrated in fig. 3. The speech signal is segmented into overlapping frames and MFCCs are computed for each frame (vectors X in the figure). The mapping function (continuous probabilistic mapping function based on GMM and described earlier) computes transformed MFCCs vectors ($\dot{Y}$). Each $\dot{Y}$ is then compared to all target database MFCC vectors in order to get the closest match $\ddot{Y}$. For each target MFCC vector, we also have memorized the corresponding auto-regressive coefficients. Thus, for each frame of the source speaker, we can get the corresponding autoregressive coefficients for the target speaker.

In parallel, for each source frame, we run a LPC analysis to extract the LP residual of the source. Finally, to synthesize converted speech, we use this residual as input excitation for the LPC synthesis filter with the auto-regressive coefficients obtained.

We know that to keep the excitation of the source is not an ideal solution since it still keeps information from the source. To approach a bit more the target voice, a pitch modification can be done on the converted speech by analysing the pitch of the source and of the target.

2) LP-based speech synthesis, with Viterbi: The voice conversion system by LPC analysis and synthesis using the Viterbi algorithm is detailed in fig. 4. As in the previous method, the mapping function transforms MFCCs from the source speech signal (X) into $\dot{Y}$ MFCC vectors. For each frame of the processed sentence, the Viterbi algorithm (that has been previously described) gives as output an MFCC vector

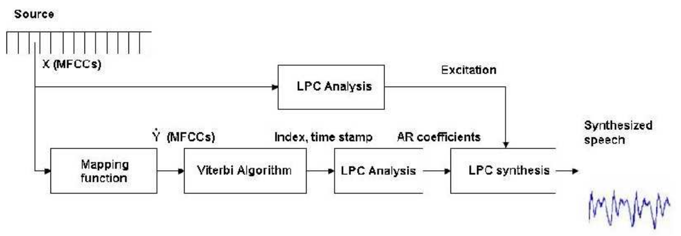

Fig. 4. Block diagram of the selection method using the Viterbi algorithm.

taken directly from the MFCC vectors of the target speech database. It also provides the index of the sentence the frame chosen comes from and the time stamp of the centre of the frame. From this information, one extracts the frame from its wave file and computes an LPC analysis to get the new AR coefficients that will be used for the synthesis.

In comparison with the previous method, one can expect an improvement in the choice of the frames in the target database, since the Viterbi algorithm takes the concatenation cost into account (and not only the target cost).

3) Speech to Speech synthesis: Instead of trying to produce the converted speech samples by LPC analysis-synthesis, as in the previous subsection, it is also possible to deliver speech by overlap-adding speech frames taken directly from the target speech files.

Here we use the natural target speech frames associated to each MFCC vector. The problem then reduces to combining these frames in order to achieve the highest quality possible. We use a simple OverLap-and-Add (OLA) procedure on the sequence of frames. However, there may be important problems associated to the discontinuities between the frames. To partially overcome this drawback, we apply a correction on the frame positions, based on the local maximization of the correlation between the already generated synthetic speech and each new frame, as used in WSOLA [9].

III. Facial expression analysis, mapping, and synthesis for face animation

Facial expression analysis and synthesis techniques have received increasing interest in recent years. Numerous new applications can be found, for instance in the areas of low bit-rate communication or in the film industry for animating 3D characters.

In this project, we are interested in head-and-shoulder video sequences, which we use to control the lip, eye, eyebrow, and head movements of a 3D talking head. First a 3D head model of the target is created (basically any 3D head will do, including a cartoon-like head). This step typically involves the extraction of Facial Definition Parameters (FDPs) from photographs of the target speaker. The system then analyzes the source video sequences and estimates 3D motion and facial expressions using the 3D head model information. The expressions are represented by a set of facial animation parameters (FAPs) which are part of the MPEG4 standard [10] (just as FDPs). The target 3D head model is then deformed according to the FAPs of the source, and new frames are synthesized using computer graphics techniques. Good examples of such video-based avatar control can be found in the work of Eisert [11].

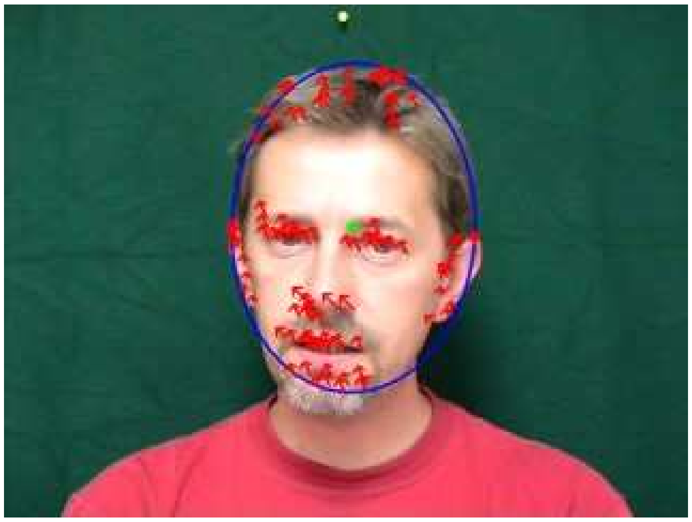

Fig. 5. An ellipse is fitted to skin blob and optical flow vectors are calculated in this region

A. Face Analysis

Face analysis process mainly consists of three tasks: detecting and tracking face region, extracting features for facial gestures from the detected face region, and tracking these features throughout the source video. In this framework, automatic face detection and computation of 8 quadratic global head motion parameters is done according to [12]. Figure 5 shows an ellipse fitted to the detected face region and optical vectors calculated between two consecutive frames that will be used in the process of computing quadratic global head motion parameters.

The global head motion parameters are used to

- estimate the position of head in the next frame,
- approximate the 3D movement of the head and
- calculate the canonical displacement vectors of facial feature points over the face region.

Once the face position is known, a set of useful feature points for face components, which are lips, eyes, and eyebrows in this scenario, are defined and tracked. For this purpose, *Active Appearance Models* (AAM) approach that was introduced by Cootes, Edwards and Taylor, is used as a means of modeling and tracking face components [13], [14], [15], [16].

AAM is, in essence, a combination of ideas from Eigenface Models [17], Point Distribution Models [18], and Active Shape Models [19] and has its roots in model-based approaches towards image interpretation named *deformable template models* where a deformable template model can be characterized as a model which, under an implicit or explicit optimization criterion, deforms a shape to match a known object in a given image [20]. Consequently, AAMs establish a compact parametrization of shape and texture, as learned from a representative training set. Thus, AAMs require model training phase before they are used to process unseen images. In this training phase, the AAM learns a linear model of the correlation between parameter displacements and the induced residuals.

During search for a face in a new frame, residuals are iteratively measured and used to correct the current parameters with respect to the main model, leading to a better fit. After a few iterations, a good overall match is obtained [15] . Applying this search method to each video frame using the model obtained after training will constitute the tracking part of the task. Figure 6 demonstrates a sequence of consecutive frames as a result of the tracking task. The output of the tracking process will be the set of 2D positions of key points of the face components for each frame which will be input to the facial movement mapping module.

B. Facial Movement Mapping

Since the previous module tracks pixel locations of the key feature points of the face components and the next module will synthesize a 3D head animation using an avatar model based on MPEG-4 parameters, the mapping function will compute MPEG-4 parameters from a set of 2D image locations.

MPEG-4 Facial Animation defines two sets of parameters: the Facial Definition Parameter (FDP) set and the Facial Animation Parameter (FAP) set [21], [22]. These two sets provide a common framework for animating a 3D face deformable mesh model with the help of high-level and low-level facial actions, closely related to facial muscle movements.

The first set of parameters, FDPs, is used to define feature points that are basic components in 3D face deformable meshes, represented by a 3D set of vertices. The movements of these vertices drive the deformations to be applied to the model to animate the desired facial expressions. 84 feature points on morphological places of the neutral head model are defined by MPEG-4 Facial Animation, as shown in figure 7.

The FDPs mainly serve for specifying how the face mesh will deform according to the transformation parameters (FAPs).

The second set of parameters, FAPs, on the other hand, consists of a collection of animation parameters that modify the positions of the previously defined feature points and, thus, can be used to create or change the model for each of the desired facial expressions. There are 68 FAPs that can be grouped in two categories: high-level and low-level parameters. The number of high-level parameters is only two. The first one is visemes, which are the visual equivalents to phonemes in speech. This parameter defines the mouth shapes produced by the different possible phonemes. The second high-level parameter corresponds to facial expressions and can take 6 values, one for each of the 6 archetypal emotions (anger, disgust, fear, joy, sadness and surprise). The remaining 66 low-level parameters are used for basic deformations applied to specific morphological points on the face, like the top middle outer-lip, the bottom right eyelid, etc... Because FAPs are universal parameters and independent from the head model geometry, MPEG-4 Facial Animation defines a set of 6 units, the Facial Animation Parameter Units (FAPU), to normalize the FAPs and make them independent of the overall face geometry. Prior to animation of a virtual face, the FAPs have to

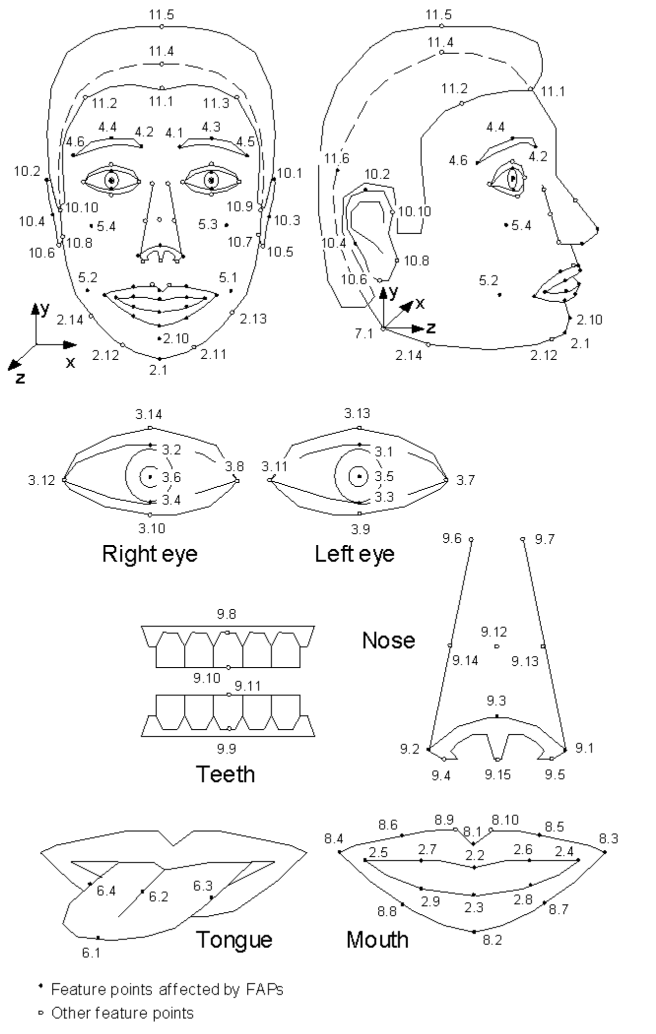

Fig. 7. Feature points are used to define the shape of a face model and the facial animation parameters are defined by motion of some of these feature points

be normalized by their corresponding FAPUs. As depicted in figure below, the FAPUs are defined as fractions of distances between key facial features (i.e. eye-to-eye distance, angular unit, etc.).

At this point, 8 quadratic global motion parameters are used to separate the local movements of key feature points from the global movement of head, since input value for each FAP is independent from other FAPs.

One can think about this scenario: if head rotates around its horizontal axis by 45 degrees, mouth will also rotate with head. In the meantime, mouth just opens a little bit, meaning that upper and lower lips move away from each other. And now, if one looks at the big picture, middle point of upper lip will seem to be moving neither just horizontally nor just vertically, but along an inclined line in between. If the deformation value for the middle point of the upper lip is calculated directly by taking the difference of the 2D pixel locations between two consecutive frames, the final value will obviously be a wrong input for the animation. Instead of this, having $(x(t), y(t))$ in frame t, for frame $t+1$, one can estimate $(\hat{x}(t+1), \hat{y}(t+1))$ using global motion parameters $(a_1, \ldots, a_8)$ and $(x(t), y(t))$.

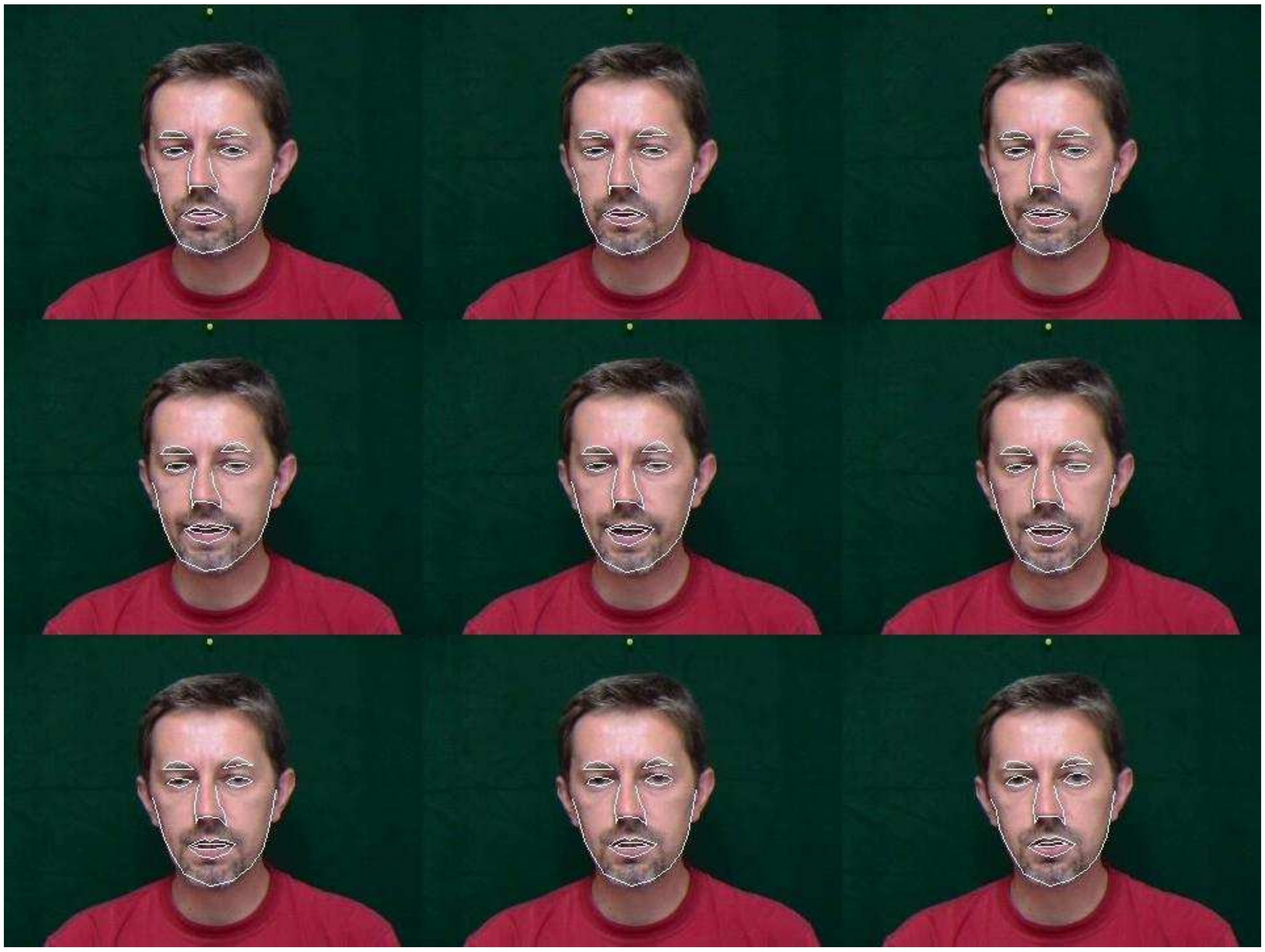
Fig. 6. Example image sequence that demonstrates the performance of tracking by AAMs

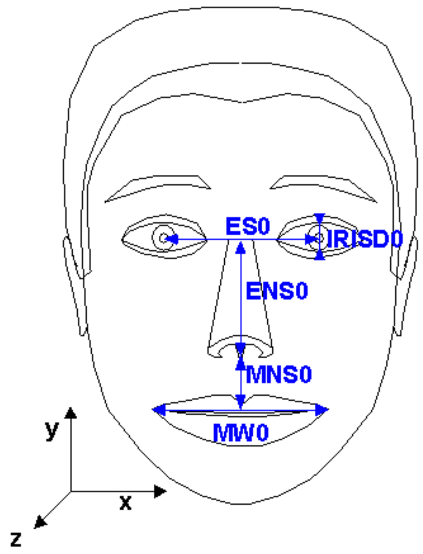

Fig. 8. Some of the feature points are used to define FAP units when the face model is in its neutral state. Fractions of distances between the marked key features are used to define FAPUs

Then, the difference between the measurement $(x(t+1), y(t+1))$ and estimate $(\hat{x}(t+1), \hat{y}(t+1))$ can be considered as the actual deformation of the model for the lips. Likewise, all the values for the used FAPs can be calculated in a similar fashion.

Besides calculating the FAP values for different face components, FAP values for the head movement itself have to be computed as three rotation angles around the three axes of the head. This is not an easy task since going from 2D world to the 3D world with only one angle of view requires sophisticated approximations, where there is no chance for accurate results at all. There are model-based pose estimation approaches using ellipsoidal models [23], or downhill simplex optimization method and the combination of motion and texture features [24], [25].

However, even though it is not as accurate as other methods, it is also possible to simply approximate the rotation angles by assuming that the head center moves around over the surface of a sphere that is centered at the top of the spinal cord. When the displacement of head is projected onto this sphere, the angles of head rotation can be estimated approximately.

IV. Experiments and Results

In order to design the application depicted in Fig. 1, we needed to choose a source and target speaker, make sure we could have access to a large amount of speech from the target (for the speech synthesis module), of a reasonable amount of aligned speech data for source and target (for the voice mapping module), and of some test speech+video data from the source (in order to test the complete system). We chose to use the CMU-ARCTIC databases as target speech [26]. This database was designed specifically for building unit-selection-based TTS systems. It is composed of eight voices, speaking 1150 sentences each (the same sentences, chosen to provide a phonetically balanced corpus). We thus decided to record an audiovisual complementary database, for use as the source data.

The eNTERFACE06_ARCTIC database we have created is composed of 199 sentences, spoken by one male speaker, and uniformly sampled from the CMU_ARCTIC database [26]. For each sentence, an .txt, a .avi, and a .wav files are available. The .avi file contains images with 320x240 pixels, 30 frames per second, of the speaker pronouncing the sentence (Fs=44100 Hz). The .wav file contains the same sound recording as in the .avi file, but resampled to 16 kHz.

The database was recorded using a standard mini-DV digital video camera. The recording of the speech signal was realized through the use of a high-quality microphone, specially conceived for speech recordings. The microphone was positioned roughly 30cm below the subject's mouth, outside the camera view.

The background consisted of a monochromatic dark green panel that covered the entire area behind the subject, to allow easier face detection and tracking. Natural lighting was used, so that some slight illumination variation can be encountered among the files (Fig. 2).

The recordings were made using the NannyRecord tool provided by UPC Barcelona, which makes it possible for the speaker to hear the sentence it has to pronounce twice before recording it. The source speaker used for the recordings were the *"awb"* speaker of CMU_ARCTIC. The eNTERFACE06_ARCTIC speaker was asked to keep the prosody (timing, pitch movements) of the source, while using his own acoustic realization of phonemes, and of course, his voice (i.e., not trying to imitate the target voice). This particular setting has made it possible for the eNTERFACE_ARCTIC recordings to be pretty much aligned with the corresponding CMU_ARCTIC recordings.

Following the standard approach, the parallel database was further divided into three subsets:

- *development* set, consisting of 188 sentences (of the total 198), used during the training phase of the different algorithms (alignment, voice mapping, etc.),
- *validation* set, used to avoid overfitting during the refinement of the model parameters (number of clusters, GMMs, etc.),
- *evaluation* set, used to obtain the results of the multi-modal conversion.

It is worth mentioning that this last subset (evaluation) was not present in any of the stages of the training. The results we have obtained can therefore be expected to generalize smoothly to any new data.

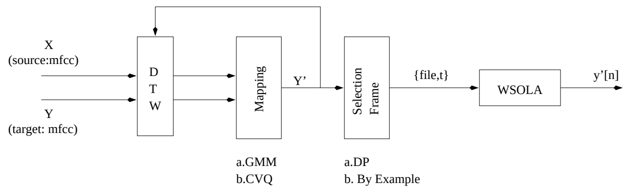

Fig. 10. Block diagram show the different alternatives for the mapping estimation and for the frame selection.

The features computed from the signals were either 13 or 20 MFCC's and their first and second derivatives; the frame rate was chosen to be 8ms. For the computation of the derivatives we relied on an implementation by Dan Ellis[2] and the other signal processing steps were taken from the MA toolbox by Elias Pampalk[3]. We also computed estimations of the fundamental frequency by using functionalities provided by the Praat software[4]. For fundamental frequency and for signal energy we can also provide first and second derivatives so that for each frame the full set of features was: 20 MFCC's, 20 ΔMFCC's, 20 $\Delta\Delta$MFCC's, f_0, Δf_0, $\Delta\Delta f_0$, energy, Δenergy and $\Delta\Delta$energy resulting in a vector of 66 dimensions.

In figure 10, we can see the different alternatives we have implemented for each of the blocks.

A. Alignment and voice mapping

After the first alignment the Euclidean distance (L_2 norm) between the source and the target MFCCs was: 1210.23. Then the GMM-based mapping was applied and the same norm was measured on the Transformed data and the Target Data: 604.35 (improvement: 50.08%). Then the Source data were transformed and a new alignment was performed. Using the new aligned data, a new mapping function was estimated and again the Source data were transformed, and again the L_2 norm between the transformed data and the Target data was measured (397.63). The process was repeated and a new measurement of the performance of the mapping function was measured using the L_2 norm (378.18).

Without iterations, we achieve a 50% reduction of distance between the target and the source data. This percentage should be higher, and provides also an information on the difference between the two speakers, and/or of the differences in the recording conditions. After some iteration we arrive to stable mapping function, with an improvement over the initial distance between the source and the target data of: 68.76%. Figure 11 shows the reduction of the distortion due to the iteration of the algorithm.

B. On clustering parameters

The incremental alignment procedure used a Gaussian Mixture Model of the source space with 128 components. This parameter remained fixed throughout the experiments. It had influence on the construction of the aligned data as well as on the mapping from source to target space, as this mapping is

[2]http://labrosa.ee.columbia.edu/matlab/rastamat/deltas.m
[3]http://www.ofai.at/~elias.pampalk/ma/
[4]http://www.fon.hum.uva.nl/praat/

Fig. 9. Facial excerpts from the eNTERFACE06_ARCTIC database.

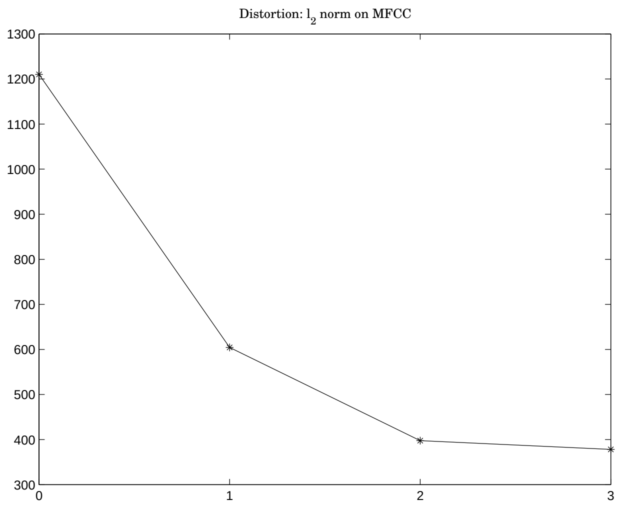

Fig. 11. Euclidean distance between the transformed data and the target data. Stability is reached after three iterations, with on overall improvement of 86%.

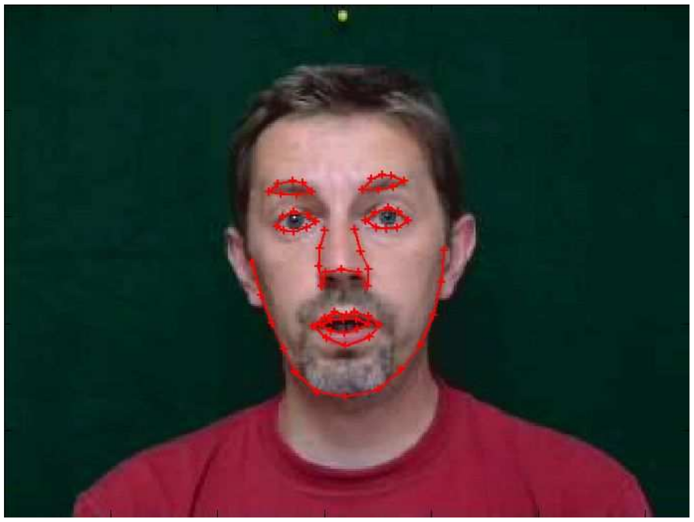

Fig. 12. Example of a training image labelled with 72 landmark points

TABLE I

MEAN SIGNAL TO NOISE RATIO FOR VALIDATION FILE FRAMES MAPPED TO SOURCE SPACE CLUSTERS

Number of Clusters	SNR
256	11.58 dB
512	12.04 dB
1024	12.46 dB

following equation 2 and so depends as well on the number of mixture components.

For the computation of the CVQ-based mapping the source space had to be clustered. Table I shows the average logarithmic signal to noise ratios for our five validation files. We can see that increasing the number of clusters brings improvements of about 1 dB. For the $Y|X$ codebook size eight was chosen, so that we had eight candidates for the $\dot{Y}$ to choose from using the Viterbi algorithm.

C. Face animation

For our specific scenario, each frame in the training set has been manually labeled with 72 landmarks by using the AAM-API [27] software (freely available for non-commercial use such as research and education). The image in figure 12 shows an example of an annotated image from the training set.

In order to obtain an accurate mapping from the 2D pixel locations to the MPEG-4 parameters, the annotation of the images in the training set should closely match the MPEG-4 FDPs. In this particular task, the mapping process includes calculation of 6 facial animation parameters units (FAPUs), besides 44 low-level FAPs. Figure 13 shows the face model used in this work, already available in the XFace project.

We have found it necessary to smooth the calculated values for animation, since the measurements in the tracking process are noisy and small scale differences in the parameters for the simulation process may have large effects in the resulting animation. Kalman Filter was used for this purpose with a state model consisting of positions and velocities of all the key feature points on the face.

2cm

As a result of using the techniques described above, several video files have been produced from the evaluation subset of the database. These results are available for tests in the archive of our project on the eNTERFACE06 website. We created parallel videos showing source speaker and target avatar side-by-side in order to evaluate the face detection and tracking algorithm. As it can be seen in the videos, both algorithms are able to provide accurate results (although they are very sensitive to the tuning of the parameters, and in some cases they result in unreliable estimations).

Utterances using the three speech generation techniques previously explained (sec. II-D) have been generated. For comparison purposes, we have also generated an English voice for the Festival Speech Synthesis System using the full CMU ARCTIC database (except the evaluation subset). The

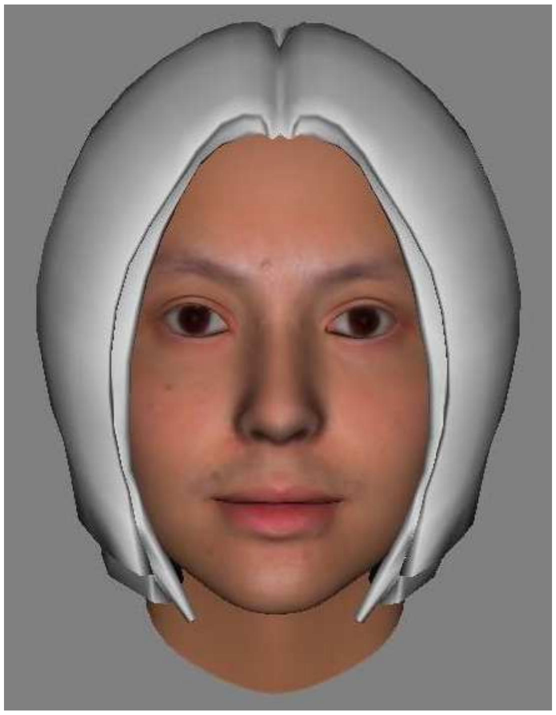

Fig. 13. The face model available in the XFace is used to create the desired animation

phonetic segmentation was performed automatically, and thus the resulting voice contains errors that would require manual correction. Informal tests show that the LPC-based methods (secs. II-D.1 and II-D.2) produce more natural, continuous sound speech, than the speech to speech synthesis method (sec.II-D.3). However, the speaker identity of the converted speech is closer in the later case to the target speaker. The highest identification score was obtained by the Festival voice, although the discontinuities due to the automatic segmentation seriously affect the quality of the synthetic speech.

V. Conclusions

In this paper we described a multimodal speaker conversion system, based on the simultaneous use of facial animation detection, 3D talking face synthesis, and voice conversion. A typical and important feature of the problem we have treated is that a large amount of speech data is assumed to be available for the target speaker.

During this work, two techniques have been studied for the mapping of the source voice to the target voice: Gaussian Mixture Models (GMMs) and Condition Vector Quantization (CVQ). Several alternatives for the synthesis block have been implemented, without achieving acceptable quality. Preliminary analysis of the results indicate that discontinuities in the phase are causing major distortion in the synthetic speech. Possible solutions and directions for future research are given below.

A. Enhanced multimodal approach

Of the two conversion problems mentioned in the introduction of this paper (voice and facial movements), we have mostly focused on the first, and assumed the second merely reduced to a scaling of source to target movements. Clearly, each human face has its own ways of producing speech, so that the facial movement conversion step could still be widely enhanced by submitting it to a more complex mapping stage.

One immediately notices that the approach followed here is only weakly multimodal: it actually uses late multimodal fusion. One of the obvious ideas that could be exploited in the future is that of using speech for face movement detection, and possibly face movements for helping voice mapping. A further step could then be to study simultaneous face and voice conversion. This would require having video data for the target (which we did not have here). The speech-to-speech component of our project could then be made multimodal in essence, by using facial movements in the definition of target and concatenation costs, for instance.

B. Weighted Euclidean distance

Until now, we have not actually used the weighted Euclidean distance, instead we simply used the basic Euclidean distance. However methods exist that allow the computation of optimal weights for such a distance. A first future improvement of our results could be to apply one of those methods.

Another way to improve this part of the system could be to compute the target and concatenation cost with other measurements such as Mahalanobis, Kullback-Leibler or Itakura distance.

C. F_0 mapping and continuity

Presently, the target speech is synthesized using the residual excitation from the source speaker. Therefore the utterance has the same prosody than the source. This is a major drawback because we think this is the main reason why the final output sounds as a third speaker situated between the source and the target.

The next step in the development of an efficient open source voice conversion system should be to create a mapping function between the two speaker's prosodies.

D. Phase continuity

In order to reduce the influence of the source speaker's voice in the final result, we would like not to use his residual excitation anymore. Indeed, this residual still contains a lot of information about him.

The OLA solution proposed in section II-D.3 can be a solution. However, this will introduce a lot of phase discontinuity (aside from the energy and pitch discontinuities which can be handled more easily). We have found no elegant solution to this problem, which requires further study

E. Pitch synchronous processing

A different approach could be to use a PSOLA algorithm to achieve the resynthesis. If PSOLA is used in the synthesis step, then the absolute value of the pitch is of lesser importance as a target for the Viterbi alignment, however its delta and delta-delta are still important. Indeed PSOLA can change the overall pitch easily, but delivers lower quality speech when the shape of the pitch curve is modified.

F. 3D Head Synthesis

While we have completed the face detection, movement tracking, and 2D to 3D mapping of this project, we did not have time to work on the 3D head synthesis aspects. In this part, the main task is to convert the set of parameters into visual animations. Animations are created using XFace interface which is an MPEG-4 based open source toolkit for 3D facial animation, developed by Balci [28]. This is a straightforward process: once the FAP values have been computed, the XFace editor (or any other suitable player) can directly synthesize them (given that the parameters file format is correct).

G. Software notes

As a result of this project, the following resources are available:

- eNTERFACE06_arctic database (video and audio). 199 sentences sampled from the CMU_ARCTIC database [5]. The video part is recorded in *avi* files (320x240 pixels), 30 frames per second, unencoded. The audio part is recorded in *wav* files, 16kHz, 16 bits per sample, unencoded.
- Conditional Vector Quantization algorithm (Matlab).
- Clustering via K-means algorithm.
- Viterbi algorithm (Matlab and C++)
- OLA implementations, with optional correlation-based correction of the window positions.
- Automatic calculation of global head motion parameters.

Acknowledgment

We want to acknowledge the CMU-ARCTIC team at CMU, for having made large speech corpora available to the research community.

We are also grateful to Prof. Antonio Bonafonte, who collaborated to the initial steps in the project design, but could finally not attend eNTERFACE'06.

We also want to thank François-Xavier Fanard, who was part of Project 3, and helped us in the design of the 3D avatar. Our thanks also to Yannis Agiomyrgiannakis for kindly providing the CVQ code and technical support.

We are also grateful to Prof. Alice Caplier (INPG Grenoble) for letting us access the facial tracking system developed at INPG, even though we have not used it in the final version of our speaker conversion system.

References

[1] Y. Stylianou, O. Cappé, and E. Moulines, "Continuous probabilistic transform for voice conversion," *IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING*, vol. 6, no. 2, pp. 131–142, 1998.

[2] Y. Agiomyrgiannakis and Y. Stylianou, "Conditional vector quantization for speech coding, to be published," Institute of Computer Science, Foundation of Research & Technology Hellas, Tech. Rep., 2006.

[3] T. Dutoit, *An Introduction to Text-To-Speech Synthesis*. Dordrecht: Kluwer Academic Publishers, 1997.

[4] L. Rabiner, "A tutorial on hmm and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, February 1989.

[5] T. Dutoit and M. Cernak, "Ttsbox: A matlab toolbox for teaching text-to-speech synthesis," in *Proc. ICASSP'05*, Philadelphia, USA, 2005.

[6] D. Suendermann, H. Hoege, A. Bonafonte, H. Ney, A. Black, and S. Narayanan, "Text-independent voice conversion based on unit selection," in *Proc. ICASSP06*, Toulouse, 2006, pp. 81–84.

[7] A. Gersho and R. Gray, *Vector quantization and signal compression*. Norwell, Massachusetts: Kluwer Academic Publishers, 1992.

[8] D. Chazan, R. Hoory, Z. Kons, D. Silberstein, and A. Sorin, "Reducing the footprint of the ibm trainable speech synhtesis system," in *ICSLP*, 2002.

[9] W. Verhelst and M. Roelands, "An overlap-add technique based on waveform similarity (wsola) for high quality time-scale modification of speech," in *Proceedings of ICASSP-93*, vol. 2, 1993, pp. 554–557.

[10] *ISO/IEC IS 14496-2 Visual*, 1999.

[11] P. Eisert, "MPEG-4 facial animation in video analysis and synthesis," *International Journal of Imaging Systems and Technology, Springer*, vol. 13, no. 5, pp. 245–256, March 2003, see also: http://ip.hhi.de/comvision_G2/expressions.htm.

[12] M. E. Sargin, F. Ofli, Y. Yasinnik, O. Aran, A. Karpov, S. Wilson, E. Erzin, Y. Yemez, , and A. M. Tekalp, "Combined gesture-speech analysis and synthesis," in *Proc. of the eNTERFACE'05 The SIMILAR Workshop on Multimodal Interfaces*, August 2005.

[13] T. F. Cootes, G. J. Edwards, , and C. J. Taylor, "Active appearance models," in *Proc. European Conf. on Computer Vision*, vol. 2, 1998, pp. 484–498.

[14] ——, "Active appearance models," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[15] G. J. Edwards, C. J. Taylor, , and T. F. Cootes, "Interpreting face images using active appearance models," in *Proc. 3rd IEEE Int. Conf. on Automatic Face and Gesture Recognition*. IEEE Comput. Soc., 1998, pp. 300–305.

[16] T. F. Cootes and C. J. Taylor, "Statistical models of appearance for medical image analysis and computer vision," in *Proc. SPIE Medical Imaging*, San Diego, CA, 2001, pp. 236–248.

[17] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proc. 1991 IEEE Com. Soc. Conf. on CVPR*. IEEE Com. Soc. Press, 1991, pp. 586–591.

[18] T. F. Cootes and C. J. Taylor, "Active shape models smart snakes," in *Proc. British Machine Vision Conf.,BMVC92*, 1992, pp. 266–275.

[19] T. F. Cootes, C. J. Taylor, D. Cooper, , and J. Graham, "Training models of shape from sets of examples," in *Proc. British Machine Vision Conf., BMVC92*, 1992, pp. 9–18.

[20] R. Fisker, "Making deformable template models operational," Ph.D. dissertation, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, 2000.

[21] M. Tekalp, "Face and 2d mesh animation in mpeg-4," *Tutorial Issue On The MPEG-4 Standard, Image Communication Journal, Elsevier*, 1999.

[22] I. Pandzic and R. Forchheimer, *MPEG-4 Facial Animation : The standard, implementation and applications*. Wiley, 2002.

[23] S. Basu, I. Essa, and A. Pentland, "Motion regularization for model-based head tracking," in *Intl. Conf. on Pattern Recognition (ICPR '96)*, Vienna, Austria, 1996.

[24] F. Preteux and M. Malciu, "Model-based head tracking and 3d pose estimation," in *Proceedings SPIE Conference on Mathematical Modeling and Estimation Techniques in Computer Vision*, vol. 3457, San Diego, CA,, July 1998, pp. 94–110.

[25] M. Malciu and F. Preteux, "A robust model-based approach for 3d head tracking in video sequences," in *Proc. of 4th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2000)*, Grenoble, France, March 2000, pp. 26–30.

[26] J. Kominek and A. W. Black, "CMU ARCTIC databases for speech synthesis," Carnegie Mellon University, Tech. Rep., 2003, http://festvox.org/cmu_arctic/.

[27] M. B. Stegmann, B. K. Ersboll, and R. Larsen, "Fame - a flexible appearance modeling environment," in *IEEE Transactions on Medical Imaging*, vol. 22, no. 10, October 2003, pp. 1319–1331.

[28] K. Balci, "Xface: Mpeg-4 based open source toolkit for 3d facial animation," in *Proc. Advance Visual Interfaces*, 2004, pp. 399–402.

[5]Full database is available at: http://www.festvox.org

Thierry Dutoit graduated as an electrical engineer and Ph.D. in 1988 and 1993 from the Faculté Polytechnique de Mons, Belgium, where he is now a full professor.

He spent 16 months as a consultant for AT&T Labs Research in Murray Hill and Florham Park, NJ, from July, 1996 to September, 1998. He is the author of two books on speech processing and text-to-speech synthesis, and the coordinator of the MBROLA project for free multilingual speech synthesis.

T. Dutoit was an Associate Editor of the IEEE Transactions on Speech and Audio Processing (2004-2006) and is a member of the INTERSPEECH'07 organization committee.

André Holzapfel graduated as an electrical engineer in 2004 from the University of Applied Sciences Duesseldorf, Germany, and will complete work on his Master Thesis at the Graduate School University of Crete in September.

He has served Creamware Datentechnik GmbH from 2002 until 2003 to develop dynamic models for electronic tubes on DSP platforms. His special interest is information retrieval from musical signals.

Matthieu Jottrand holds an Electrical Engineering degree from the Faculté Polytechnique de Mons since June 2005. He did his master's thesis in the Image Coding Group of Linköping Institute of Technology. Matthieu is a researcher fromTCTS lab (FPMs) since September 2005.

He is currently working in the field of ASR for the IRMA project (development of a multimodal interface to search into indexed audiovisual documents) and just started a PhD thesis under the supervision of Thierry Dutoit.

Ferran Marqués is Professor at Technical University of Catalonia (UPC), Department of Signal Theory and Communications, Barcelona, Spain. He received a degree on Electrical Engineering in 1988, and the Ph.D. degree in 1992, both from the UPC.

From 1989 to 1990, he joined the Digital Image Sequence Processing and Coding Group at the Signal Processing lab. of the Swiss Federal Institute of Technology (EPFL). In June 1990, he joined the Department of Signal Theory and Communications of the Technical University of Catalunya (UPC). From June 1991 to September 1991, he was with the Signal and Image Processing Institute (SIPI) at the University of Southern California (USC). In 1993 he joined the Technical University of Catalonia where he is currently Professor.

He has been President of the European Association for Signal, Speech and Image Processing (EURASIP) in the term 2002-2004. He served as Associate Editor of the Journal of Electronic Imaging (SPIE) in the area of Image Communications (1996-2000), as member of the EURASIP Journal of Applied Signal Processing Editorial Board (2001-2003) and of the Hindawi International Journal of Image and Video Processing (2006-).

Alexis Moinet holds an Electrical Engineering degree from the FPMs (2005). He did his master thesis at the T.J. Watson research Center of IBM (2005). He is currently working on the IRMA project in the Signal Processing Lab of the FPMs. He is particularly interested in glottal source/vocal tract decomposition of speech and music, phase vocoder, voice activity detection, voice conversion and HMM synthesis.

Ferda Ofli received B.Sc. degree in Electrical and Electronics Engineering, and B.Sc. degree in Computer Engineering from Koç University, Istanbul, Turkey in 2005. He is currently a M.Sc. student in Electrical and Computer Engineering Department and a member of Multimedia, Vision and Graphics Laboratory at Koç University.

He is currently taking part in European projects, SIMILAR NoE and 3DTV NoE. His research interests include image and video processing, specifically, object segmentation and tracking, human body modeling, motion capture and gait/gesture analysis.

Javier Pérez Mayos received his M.S degree in Electrical Engineering from the Royal Institute of Technology (KTH), Stockholm, Sweden, in 2001, and from the Technical University of Catalonia (UPC) in May 2002.

Since May 2002, he is a PhD student at UPC. His research topic is voice source analysis and characterization. The objective is to be able to use this information in voice generation algorithms, so applications like emotional and expressive synthesis, and voice conversion, can benefit from his research.

He has participated in several international speech-to-speech translation projects (LC-STAR, TC-STAR) and has released Gaia, a research-oriented speech-to-speech translation architecture.

Yannis Stylianou is Associate Professor at University of Crete, Department of Computer Science. He received the Diploma of Electrical Engineering from the National Technical University, NTUA, of Athens in 1991 and the M.Sc. and Ph.D. degrees in Signal Processing from the Ecole National Superieure des Telecommunications, ENST, Paris, France in 1992 and 1996, respectively.

From 1996 until 2001 he was with AT&T Labs Research (Murray Hill and Florham Park, NJ, USA) as a Senior Technical Staff Member. In 2001 he joined Bell-Labs Lucent Technologies, in Murray Hill, NJ, USA. Since 2002 he is with the Computer Science Department at the University of Crete.

He was Associate Editor for the IEEE Signal Processing Letters from 1999 until 2002. He is Associate Editor of the EURASIP Journal on Speech, Audio and Music Processing. He was served on the Management Committee for the COST Action 277: "Nonlinear Speech Processing" and he is one of the two proponents for a new COST Action on Voice Quality Assessment. He holds 8 patents. Prof. Stylianou participates in the SIMILAR Network of Excellence (6th FP) coordinating the task on the fusion of speech and handwriting modalities.