Voice transformation algorithms with real time DSP Rapid Prototyping tools

Version 1.0

Graziano Bertini, Federico Fontana, Diego Gonzalez, Lorenzo Grassi, Massimo Magrini

Technical Report -

ISTI-CNR

Voice transformation algorithms with real time DSP Rapid Prototyping tools

Graziano Bertini*, Federico Fontana**, Diego Gonzalez**,
Lorenzo Grassi***, Massimo Magrini*,

*C.N.R. Institute of Science and Information Technology, Pisa, Italy
**CNR Fondazione Scuola san Giorgio , Venice, Italy
*** ASPER srl, Florence, Italy

Keywords: Digital Signal Processing, Speech Analysis and Synthesis, Audio Restoration, Rapid Prototyping

ACM Classification: C.3 Signal Processing and Systems, H.5.1 Audio Input/Output, H.5.5 Sound and Music Computing, I.5.4 Waveform Analysis

Contents

1 Abstract

The main goal of the work here described is the DSP implementation of innovative algorithms for voice transformation in real time.

This work represents part of the procedure conceived for the reconstruction of voice and dialogue in audio tracks of old and highly damaged film movies that has been developed within the framework of the RACINE-S European Project.

In addition to the implementation of a set of methods, as LPC/ VC (Linear Prediction Coding/ Voice Conversion) techniques, innovative non-linear algorithms for improving the quality of synthesized speech have been proposed.

This entire set of operations represents a particular implementation of the so-called "Virtual Dubbing" procedure. The main steps of the complete project include: a method for high-quality voice transformation, designing a suitable algorithm in Matlab/Simulink and, finally, translating it into Digital Signal Processor target code by means of a rapid prototyping approach.

The original code was developed on Matlab and so we have used the Mathworks MATLAB's Real Time Workshop (RTW) DSP platform for rapid prototyping.

In fact, with the advent of the Real Time Workshop, is now possible to compile, load and execute graphically designed Mathworks Simulink models into a real DSP platform. RTW supports many Developer Starter Kit DSP boards, including the pretty fairly Texas Instruments C6000 series that has been used for the present work.

2 Introduction

With the term Audio restoration we normally refer to the process of improving the listening pleasure or intelligibility of recorded sound by using equalization, phase optimization, noise removal and other techniques. An example of a common audio restoration process is removing background noise from old musical recordings: impulsive (thumps, scratches, ticks, clicks) or non-impulsive (hiss, hums, vinyl surface noise from etc.).

Many different techniques have been developed for enhancing the quality of old and/or very deteriorated photographic images, films, audio and musical recordings.

Nevertheless there are still very active research areas involved with the development of special applications for solving non-standard problems, as in our studied case. The RACINE-S Project involves both image (not described in this document) and synchronized audio restoration. In our case we are managing highly deteriorated or even missing parts of the audio score, as happens very often in very old movies. For this reason we are talking in this case about "reconstruction" rather than "restoration".

Several partners worked together in the Project [1] for investigating and developing techniques for restoring old and very deteriorated movies. The Project involves different research/university units and industrial partners, each working on a different aspect: image/audio reconstruction and audio rendering improvement using intensimetric acoustic ambience reconstruction.

In this document we will describe only the part of the work related to the voice reconstruction of the audio sequences. This work is developed under the

responsibility of the Musical and Architectural Acoustics Lab. of FSSG-CNR of Venice (I).

3 Project overview

The starting point for virtual dubbing consists of digitized audio data arising from archive movie's sequences which are highly damaged or even partially missing. In this last case information are taken from segments positioned before and after the missing pieces of film. This audio data, which are supposed to be available as standard audio file, contains the relevant information about the target voice which needs to be reconstructed.

In a traditional dubbing studio a professional dubbing speaker dubs the sentences spelled by the target voice, if available, or simply reads the content of the script, if the corresponding sequence is missing. The main idea at this point is to provide the voice of the professional dubbing speaker with the features of the target voice. This *voice conversion* process, described later, can be done in real time at the dubbing process or off-line at an audio post-production level (the most probable case in presence of missing sequences). This process is performed in two phases: *analysis* and *synthesis*.

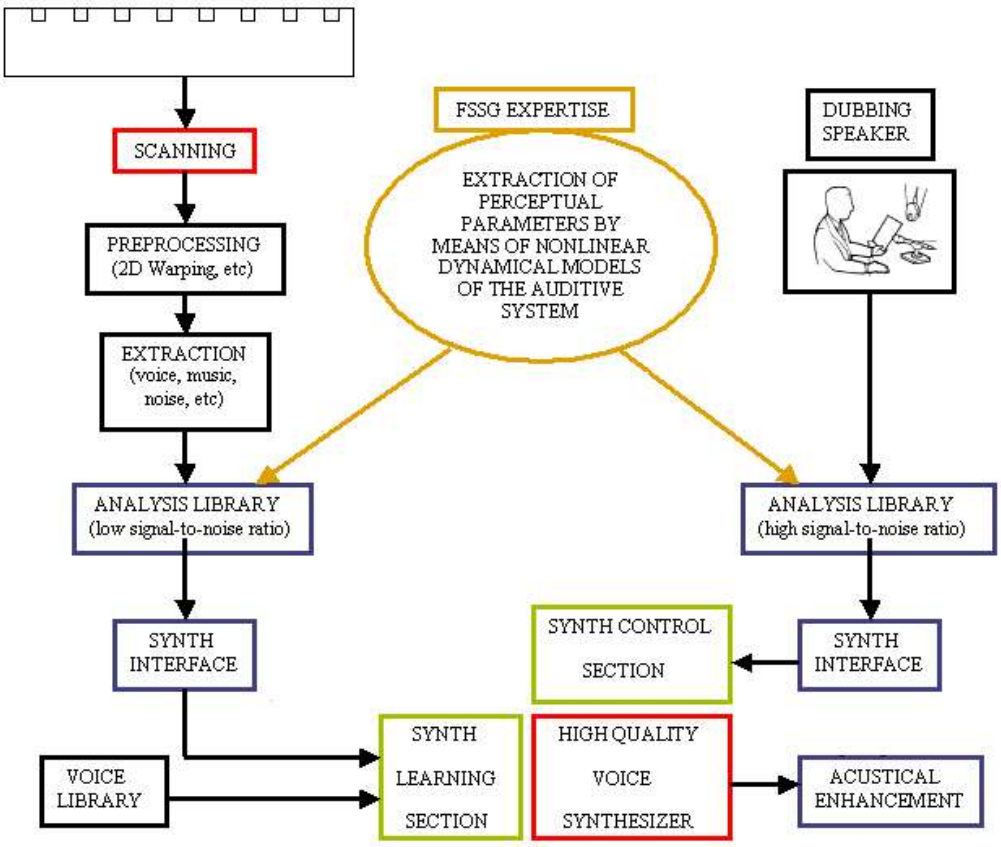

Fig. 1 Virtual dubbing block diagram

An analysis sub-process is performed on both target and source (the dubbing speaker) voices. The synthesis process arises later: the live or recorded voice of the dubbing speaker takes control over the semantics of the high quality voice synthesizer in real time, while the characteristics of the target voice are taken from a library representing the learning of the "transformation functions" by the synthesizer (fig.1).

In this document we will describe how this idea has been implemented, using a rapid prototyping approach supporting state-of-the art DSP technology.

For this reason it involves several different aspects, ranging from deep mathematical modeling to DSP hardware knowledge, and it has been splitted into different sub-tasks carried out by different working units.

3.1 Matlab/Simulink modeling

The basic algorithms have been developed by FSSG-CNR under the Mathworks'MATLAB environment, using firstly the common scripting interface, and later translating into Simulink models. These *Voice analysis and synthesis* algorithms, based on LPC (Linear Prediction Coding) methods plus special non-linear techniques, have been deeply tested and simulated before passing them to the DSP implementation phase.

3.2 DSP implementation

Because the complete implementation of the method involves a high algorithmic complexity, for practical applications is convenient to test the possibility to implement it in real-time. For this reason an implementation of the algorithm on a state-of-the art DSP platform has been tested. As the basic algorithms are being developed at a research level, such a test needs to be performed in a flexible platform which allows the implementation of the non-optimized algorithms with a reasonable effort, without the need of a time consuming "source code" porting.

For this task Mathwork's Real Time Workshop (RTW) has been used. Using RTW the previously simulated Simulink models have been automatically translated into DSP source code, and then compiled and loaded on the DSP board.

3.3 Additional HW development

For an easier use of the DSP starter kit board, a special enclosing hardware with additional analog circuitry has been built. This embedding hardware system provides input/output interfacing (analog and digital), power supply, signal conditioning (low noise preamplifier) and a user panel with state monitoring and reset buttons. The enclosing case also shields from electromagnetic interferences, improving the system reliability. A more detailed description can be found on [2].

Fig.2 shows the enclosing HW realized for the laboratory testing, connected to the PC host.

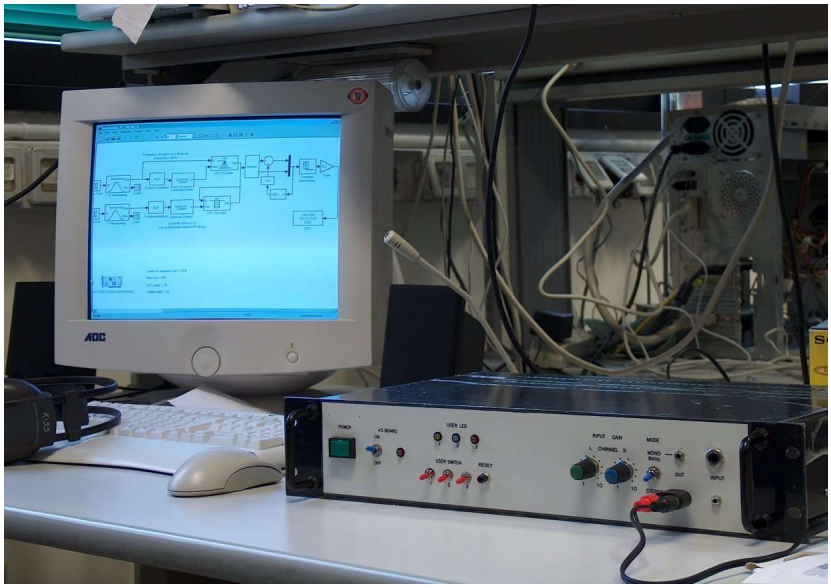

Fig. 2 Embedding hardware prototype.

4 Voice analysis and re-synthesis by LPC-VC method

We can distinguish between two main categories of speech generation techniques: rule-based synthesis and concatenative synthesis: The former relies on rules controlling the parameters of an articulatory model; the latter builds up words and sentences by connecting segments of speech selected from a database.

Alternatively, we can distinguish between time-based techniques (e.g. PSOLA and its variants[3-4]) and frequency domain representations such us the Harmonic plus Noise Modeling, HNM [5-6-7-8]. Every method has its advantages and drawbacks, and very often the choice depends on application's constraints.

Among the time domain methods the LPC (Linear Prediction Coding) encoding provides good compression ratio and simple time and pitch scaling procedures.

As said before our project requires the conversion of a *source* speaker's voice into another, as if it were pronounced by a different (*target*) speaker.

Today's available *voice conversion algorithms* mainly rely on residual-excited LPC synthesis [9-10-11] or on STFT-based synthesis [12-13].

In both cases the core of the algorithm is the definition of a map *F(x)* which transforms a feature vector *x* onto a new vector *y*. In the former case the feature vector is typically a set of LPC coefficients or, due to their better interpolation properties, a set of line spectral pairs coefficients (LSP) [9]. An example of such a voice conversion algorithm is available in the FESTIVAL TTS system within the OGIresLPC synthesis plug-in. In the latter case the feature vector can be some sort of spectrum magnitude representation, such as the FFT magnitude or the mel-cepstral coefficients [12].

A simple definition of the map F can rely on some parametric non-linear function approximation tools, e.g., neural networks or radial basis function networks (RBFN) [13]. Another common approach to the definition of the map structure is to use a probabilistic, locally linear function. A widely used technique makes use of Gaussian

Mixture Models (GMM), which are capable to embed in the mapping function an acoustic model of the source speaker based on a Gaussian mixture [9,12].

The design of the conversion function requires a number of minimal steps. First, a database of sentences uttered by different speakers must be generated. Then, the feature vectors are computed by a frame-based LPC or sinusoidal analysis. Before the training step a dynamic time warping procedure (DTW) is required to time-align the feature vectors derived from the first speaker with the ones accounting for the second speaker. This guarantees that a input-output training is worked out, in which a phoneme of the first speaker corresponds to the same phoneme of the second speaker. At this stage, a conversion function based on a RBFN can be trained directly by identifying the parameters given by the input-output training pairs.

Another strategy to create a conversion function, again embedding an acoustic model of the speaker, first requires to build the GMM modeling the probability density function (pdf) of the source speaker, then to identify the remaining parameters of the conversion function. The computation of the acoustic model is usually made by training the GMM with an expectation-maximization algorithm (EM). After the training, the conversion function will completely represent the new voice given the source voice.

4.1 Voice conversion model

In our project we use a voice conversion model in which two training sets of LPC coefficients extracted by the time window series and containing the source and the target (voice) signal, respectively, are used to form the conversion function (fig.3). This function is then used to map source into target LPC coefficient sets.

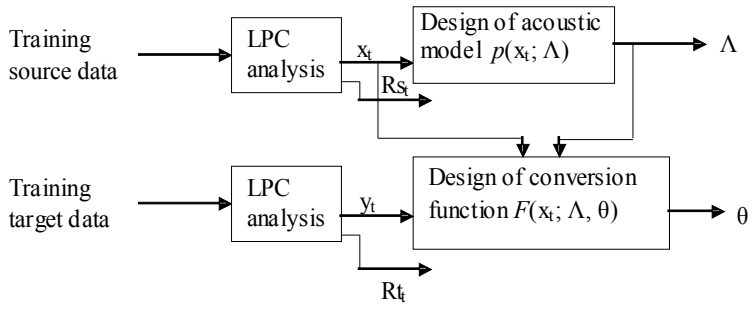

Fig. 3 Acoustic model design.

The reasons for such a choice are supported by the following arguments:

- Although today's state-of-the-art voice conversion strategies strongly rely on signal-based techniques, such as HNM, model-based design is nevertheless emerging in several application fields for its inherent ability to describe physical phenomena directly. Model-based systems shift the focus to the physical systems, treating the signals they generate as natural consequences of their dynamics. It is likely that the more accurate a system description, the more competing the model performance is compared to traditional signal-based structures.

- Accuracy in the model description hardly ever means that any subtlety of the physical system has to be modeled as carefully as possible. Rather, model-based design demands that only the phenomena and the events that are crucial in the voice production process are precisely reproduced. From this perspective, LPC has long been recognized to represent a successful approach to the physical modeling of the vocal tract [14].

- The advantage of choosing the model-based approach lies in the accessibility to the system control, that is, the physical meaningfulness and directivity of the control parameters. Apart from a generally more intuitive structure of the system itself which is made possible by the pregnancy of its block diagram and related controls, the main pro coming from the aforementioned feature lies in the inherent flexibility of the access layer of the model: this flexibility enables model-based systems to be interfaced each other directly, due to an inherent matching of the input/output quantities incoming and outgoing to and from the models.

- For the purpose of this project, the high interconnectivity of model-based systems fosters a combination of the LPC voice conversion block with a novel family of non-linear glottal models, especially designed for the generation of source signals which are typically produced in the human glottis: interfacing the two blocks will result in an overall structure corresponding to the human vocal tract which is composed by an excitation and resonant structure, here coherently modeled by interconnecting the glottal and LPC (vocal tract) models.

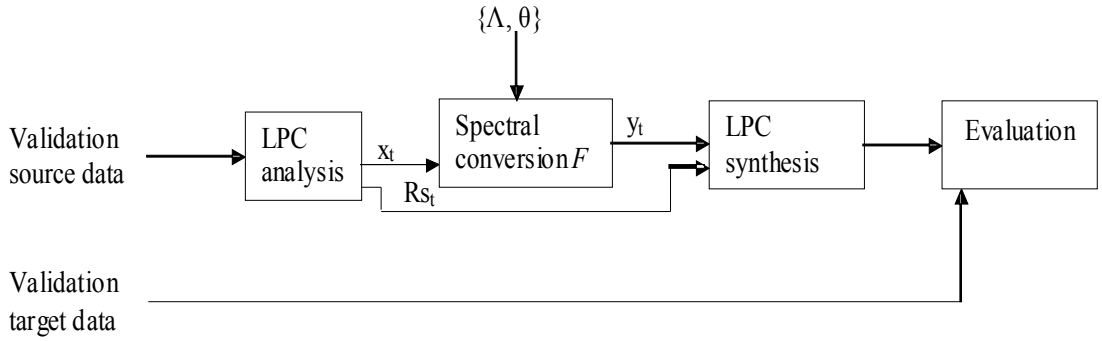

Fig. 4 Voice conversion engine.

Our LPC-VC model synthesizes the target voice by filtering the LPC residual of the source voice using target LPC coefficients obtained by converting the corresponding source LPC parameters by means of F (fig.4). A careful design of the LPC-VC model would require to segregate the voice signal in “voiced” and “unvoiced” segments, each contained in a separate time window. Segmentation is currently studied at different levels and, although automatic segmentation systems exist aiming at dividing the voice signal optimally, it is often worked out manually especially in high-end voice synthesis applications. At this stage of the project a standard segmentation of the signal into small segments having equal time length is sufficient to test the LPC voice conversion.

4.2 Conversion functions based on GMMs

The GMM approach assumes that the density of an observed process can be modeled as a weighted sum of component densities:

$$f(\vec{x}/\Lambda)=\sum_{i=1}^{M}\alpha_i N(\vec{x};\vec{\mu},\Sigma_i) \qquad -1$$

where [x\vec] is a P-dimensional input vector, N([x\vec];[(μ_i)\vec],Σ_i) are the component densities, and α_i are the mixture weights. Each component density is a P-variate Gaussian function of the form

$$N(\vec{x};\vec{\mu},\Sigma_i)=\frac{1}{(2\pi)^{p/2}}\left(|\Sigma_i|\right)^{-1/2}e^{-1/2(\vec{x}-\mu_i)^T\Sigma_i^{-1}(\vec{x}-\mu_i)} \qquad (2)$$

with mean vector [(μ_i)\vec] and covariance matrix Σ_i. The weights α_i satisfy the constraints $\sum_i^M\alpha_i=1$ and $\alpha_i \geq 0$.

The Gaussian mixture is completely specified by the mean vectors, covariance matrix and mixture weights, and can be represented by

$$\Lambda=\{\alpha_i,\mu_i,\Sigma_i\};i=1...M \qquad (3)$$

An interesting feature of the GMM for sound processing applications is that the component densities of the mixture may represent a partition of the underlying sound process in a set of acoustic classes. The probability that an observed input vector [x\vec] belongs to the class $\lambda_i=(\alpha_i, [(\mu_i)\vec], \Sigma_i)$ is given, in terms of density, by the formula

$$p(\lambda_i/\vec{x})=\frac{f(x/\lambda_i)p(\lambda_i)}{f(x/\Lambda)}=\alpha_i\frac{N(x;\mu,\Sigma_i)}{\sum_{j=1}^{M}\alpha_j N(x;\mu,\Sigma_i)} \qquad (4)$$

where f([x\vec]|λ_i) = N([x\vec];[(μ_i)\vec],Σ_i), p(λ_i)=α_i, and f([x\vec]|Λ) is given by Eq. (1).

When used to model the speech process, the components of the GMM represent different phonetic events. Let us suppose that a sequence of P-dimensional column vectors {[x\vec]$_t$}, t=1,...,T, which represents the time-varying spectral envelope of a source signal, has been fitted by a GMM. Moreover, let assume that a sequence of P′-dimensional column vectors {[y\vec]$_t$}, t=1,...,T, having the same length of the source signal, is the target of the conversion. We define a spectral conversion function as a map F:\mathbbR^P→\mathbbR$^{P'}$ able to transform each vector in the input sequence into the vector which occupies the same position in the output sequence, thus preserving the time information of the input and output data. Although it is not

necessary for the input and the output vectors to have the same dimension, we will assume P′=P in the rest of the paper. A common way to define a conversion function is to use a parametric form for the spectral conversion function [12]:

$$\Im(\vec{x}_t) = \sum_{i=1}^{M} p(\lambda_i / \vec{x}_t)\left[\vec{\theta}_i + \Gamma_i \Sigma_i^{-1}(\vec{x}_t - \vec{\mu}_i)\right] \tag{5}$$

This conversion equation is equivalent to the solution of the following set of equations:

$$\vec{y}_t = \sum_{i=1}^{M} p(\lambda_i / \vec{x}_t)\left[\vec{\theta}_i + \Gamma_i \Sigma_i^{-1}(\vec{x}_t - \vec{\mu}_i)\right] \tag{6}$$

for all t=1... T. Eq. (6) can be gathered into a single matrix equation by:

$$\mathbf{Y}=\mathbf{P}\cdot\Theta + \Delta\cdot\Gamma \tag{7}$$

where

$$Y = \left[y_1 \ldots y_T\right]^T \tag{8}$$

$$P = \begin{bmatrix} p(\lambda_1 / x_1) & \ldots & p(\lambda_M / x_1) \\ \ldots & \ldots & \ldots \\ p(\lambda_1 / \vec{x}_T) & \ldots & p(\lambda_1 / \vec{x}_T) \end{bmatrix} \tag{9}$$

Δ is a matrix that depends on the conditional probabilities,

$$\Theta = \left[\vec{\theta}_1 \ldots \vec{\theta}_M\right]^T$$

and

$$\Gamma=[\Gamma_1 \ldots \Gamma_M]^T$$

are the unknown parameters of the conversion function.

We omit here the term $\Gamma_i\Sigma_i^{-1}([x\backslash vec]_t - [(\mu)\backslash vec]_i)$ in (5) and we use the following reduced form of the conversion function [12]:

$$\Im(\vec{x}_t) = \sum_{i=1}^{M} p(\lambda_i / \vec{x}_t)\left[\dot{\theta}_i\right] \qquad (10)$$

4.3 Improvement of the re-synthesized voice quality.

As regards the hearing system different fundamental facts about sound perception have been described by means of non-linear techniques. One of this is the pitch perception of complex sounds.

In fact, some kinds of sounds seem to elicit psycho-acoustical responses [15] which correspond to frequencies not really present in the stimulus as in the case of sounds with the fundamental suppressed (missing fundamental). These are at the basis of a series of phenomena known as the perception of the "residue" or virtual pitch which are particularly important for understanding the perception of voice and the sound of some musical instruments. A theory capable of an accurate explanation of the experimental data about residue perception has been recently formulated [16-17]. It is based on the technique of qualitative modeling, assigning a particular parameter of the *dynamical attractor* of a generic non-linear system to the perception of pitch [18].

The qualitative modeling consists in the identification of the parameters of the dynamical attractor with the parameters describing the perceptual quality. If this kind of modeling is successfully achieved, very important goals can be accomplished from a practical point of view:

- the system becomes an efficient emulator of the perceptive function which can be implemented at a software and/or hardware level
- the low-dimensional set of parameters characterizing the perceptual process offers a sound basis for classifying and manipulating the information: a great dimensionality reduction is indeed performed
- the system is robust, that is immune to signal or system noise, because the same happens for the dynamic's properties (universality property)
- simple models can display complex behavior and can describe complex signals (deterministic chaos property)

5 Matlab/Simulink algorithms implementation

Following the Rapid prototyping approach we implemented our algorithms using Simulink.

Simulink is a software package running inside Matlab for modeling, simulating, and analyzing dynamic systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multirate, i.e., have different parts that are sampled or updated at different rates.

Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click-and-drag mouse operations.

Once a model has been defined, it is possible to simulate it. Using scopes and other

display blocks, the simulation results can be viewed while the simulation is running. The simulation results can be put in the MATLAB workspace for post processing and visualization.

The voice signal is first divided in frames having a strong overlapping factor in order to prevent from the birth of major artifacts occurring during the transition from one frame to another. For our specific purpose, here we have chosen a frame size of 1024 samples and a hop-size of 64 samples, along with a Blackman windowing that provides smooth transitions along adjacent frames: N=1024; hop=64; . . . F = enframe (s,blackman(N),hop); These figures seem to be widespread enough to provide sufficiently accurate results in the majority of the test cases. Then, each frame is LPC-encoded. An LPC order equal to 10 is considered general-purpose enough in most literature.

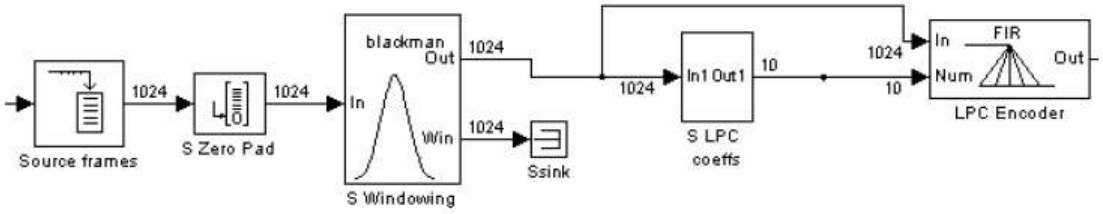

Fig. 5 Time windowing plus LPC encoding Simulink blocks.

The corresponding chain of Simulink blocks is depicted in figure 5. In this picture it can be seen that a source signal is first rearranged into overlapping frames each sizing 1024 samples, plus possible zeros padding the last frame when the overall length of the source signal is not a multiple of the frame length. More specifically, overlapping is automatically provided by the first block. Then, each frame is passed through a Blackman window stage, and every windowed frame is analyzed in order to extract the LPC coefficients. These coefficients are finally fed to the LPC encoder along with the signal to be encoded, to compute the residue.

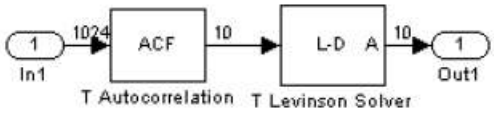

Fig. 6 LPC extraction algorithm in Simulink.

The LPC analysis is embedded in a Simulink submodule. This submodule is disclosed in figure 6. From that figure it can be seen that a classic extraction algorithm is employed, that performs an autocorrelation of the input signal followed by a Levinson-Durbin LPC extraction algorithm.

As previously said, the design and the training of the VC model are left off line, with a number of Gaussian Mixture Models (GMM's) set to 10.

The design phase first needs the target voice LPC coefficients, that are treated the same way as before.

These coefficients, transformed into line spectral pairs, are then used for the GMM modeling procedure, obtaining a set of transformation parameters (M, V, W, TH0), that will be used for the re-synthesis process.

The core steps of the re-synthesis process are:

- *lpcar2ls* converts LPC (that are autoregressive, or, briefly, AR) coefficients into line spectral pairs. This requires a *roots computation* and a *deconvolution.*

- *lpcN_fun* and *lpcpi_fun* compute parameters for calculating the set of line spectral pairs (lsp) encoding the transformed voice according to formula 10.

- the transformed line spectral pairs are finally converted back to (transformed) LPC coefficients, by *lpcls2ar*. Though, if the conversion recognizes the actual input LPC coefficient set not to belong to any category encoded by the VC model, then a transparent decoding (i.e., using a decoding coefficient set equal to [1 0 ... 0]) is chosen. In practice, during our tests we have never experienced a situation in which no category is selected for a coefficient set.

At an earlier stage, these blocks were built in Matlab functions and scripts, then they were re-implemented using Simulink blocks only. This because Matlab scripts cannot be translated into DSP code.

Among such blocks *lpcar2ls* was certainly the hardest to re-implement. Remarkable to say, no one of the above described functions was available even in the latest Simulink version. In this sense this apparently mere translation activity has a strong point, instead, in the originality of the results especially concerning the routing for the computation of the root loci.

The roots computation was resolved by a completely different numerical procedure: its solution was instead found by using the Laguerre's polynomial roots computation. This method finds the roots one by one, by iterating an estimated solution on the actual polynomial obtained by the previous (higher-order) polynomial after one of its roots has been in turn computed.

The preliminary *deconvolution* Matlab function was substituted with an embedded procedure written to perform the desired polynomial deconvolution with the given voice conversion parameters.

The whole algorithm is depicted in fig 7.

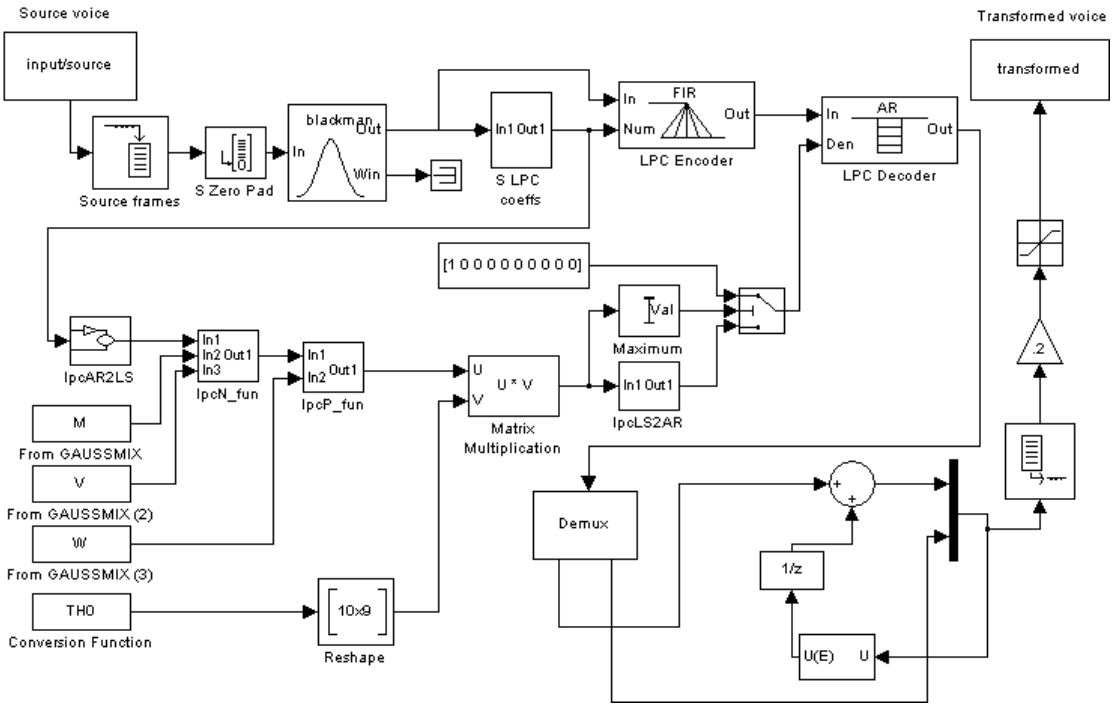

Fig. 7 Complete LPC-VC procedure in Simulink.

6 DSP Software implementation

These algorithms represent an important improvement over usual LPC techniques because through the voice transformation algorithms a detailed description of the spectral properties of the target and source signals is taken into account. Because the complete implementation of the method involves a high algorithmic complexity, for practical applications is necessary to test the possibility to implement it in real-time. As the basic algorithms are being developed at a research level, such a test need to be performed in a flexible platform which allows the implementation of the non-optimized algorithms with a reasonable effort. For this reason has been choice an implementation on a DSP hardware platform of Texas Instruments mounting one processor of the TMS320c6000 family, briefly described below, being the DSP software developed through a rapid prototyping approach. In fact, by means of the software libraries furnished by the Mathworks and Texas Instruments, is possible in principle to achieve this goal using a general methodology known as “Rapid Prototyping”.

6.1 DSP platform

In figure 8 the internal architecture of the Texas Instruments TMS320C6711 processor is depicted. The 6711 works with 32 bits registers, can address 4 Gbytes, supports big and little Indian numeric formats and can perform until 900 million floating point instructions in a second.

Moreover, the processor has many integrated peripherics, such as synchronic and asynchronic memory direct interface, host interface, external memory interface,

multi-channel buffered serial ports, IEEE-1149.1 support through JTAG and autonomous clock generation by integrated PLL, between others.

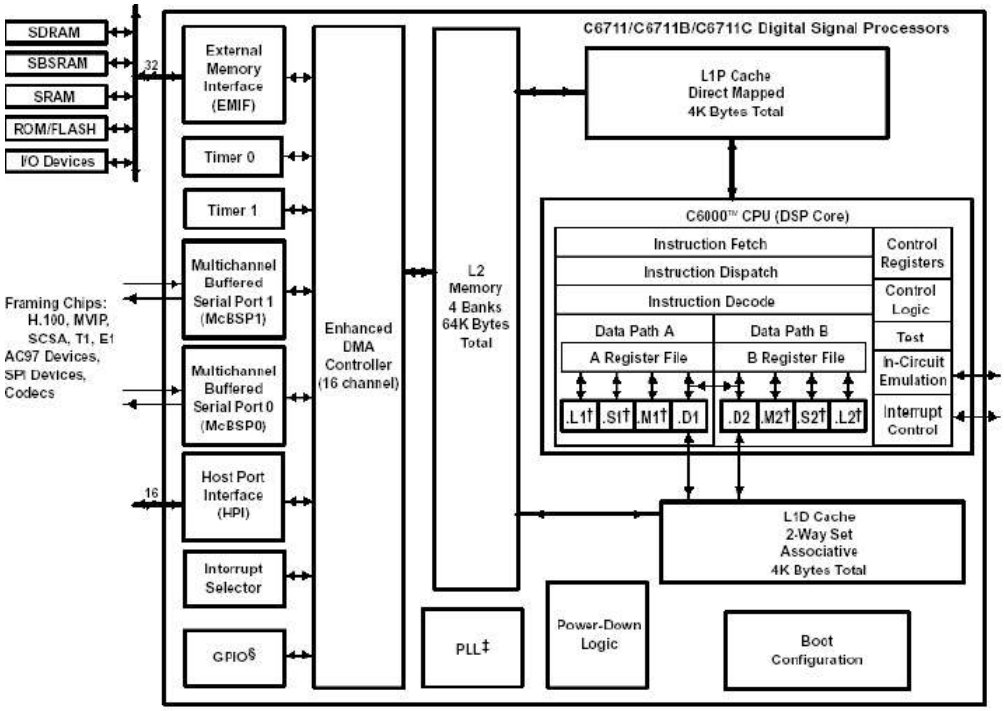

Fig. 8 internal architecture of the 6711 DSP processor from Texas Instruments

The C6000 class processors have been realized following the VLIW (very-long instruction words) technology: the CPU fetches advanced very-long instruction words (VLIW) (256 bits wide) to supply up to eight 32-bit instructions to the eight functional units during every clock cycle. The VLIW architecture features controls by which all eight units do not have to be supplied with instructions if they are not ready to execute. The first bit of every 32-bit instruction determines if the next instruction belongs to the same execute packet as the previous instruction, or whether it should be executed in the following clock as a part of the next execute packet.

Even if a discussion on the VLIW approach is beyond the scope of this paper, we remind here that a VLIW implementation has capabilities very similar to those of a super scalar processor—issuing and completing more than one operation at a time—with one important exception: the VLIW hardware is not responsible for discovering opportunities to execute multiple operations concurrently. For the VLIW implementation, the long instruction word already encodes the concurrent operations. This explicit encoding leads to dramatically reduced hardware complexity compared to a high-degree super-scalar implementation of a RISC or CISC. In this way the compiler has a much more important role, respect to super-scalar architecture, because it is fully responsible of a correct instruction scheduling. For this reason it is recommended to use the factory software tools, better with an “high level” language (e.g. C/C++). The big advantage of VLIW, then, is that a highly concurrent (parallel) implementation is much simpler and cheaper to build than equivalently concurrent RISC or CISC chips.

6.2 DSK Board

For a fast evaluation of the TMS320C6711 processor a Developer Started Kit is available from Texas Instruments (fig.9). This board have be connected to a standard PC running its development environment, Code Composer Studio.

On board there are 16 MBytes of SDRAM installed, plus 128 Kb of external Flash.

Along other features, the most relevant one, for us, is the presence of an audio codec (AD/DA converter), the TLC320AD535. Even if it's quality is rather poor (16 bit, 8 kHz sampling rate), it is sufficient to test DSP applications in the speech audio band. Anyway the board can mount additional daughter boards, as better performance A/D converters: for example the PCM3003 daughter board, which allows 48 kHz sampling rate (16 bit stereo).

A JTAG connector, for a deep control and debugging is also present on board.

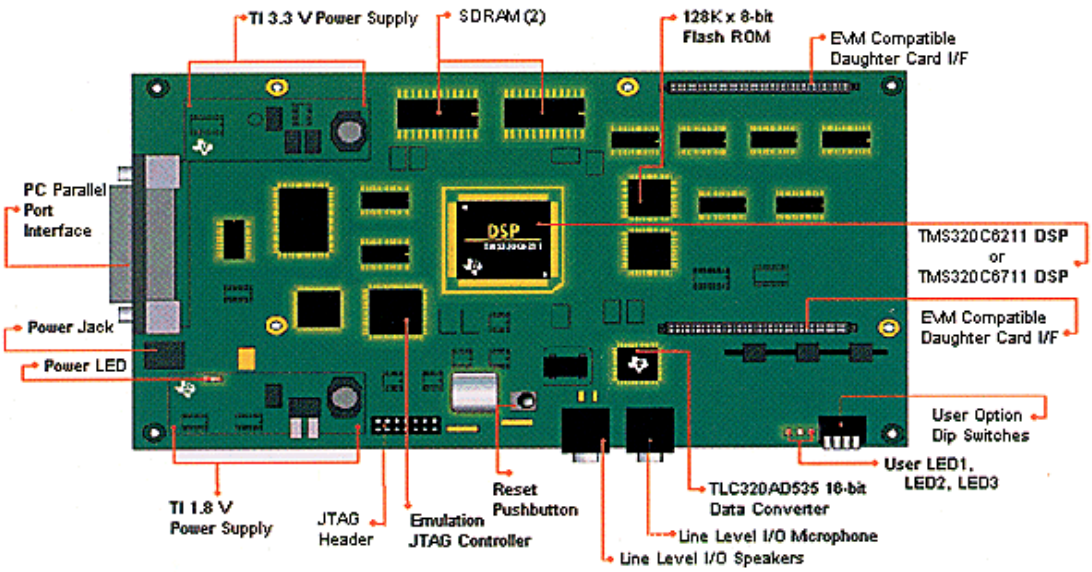

Fig. 9 view of the TMS320C6711 DSK board.

6.3 DSP Code Implementation using Real Time Workshop

Mathworks' Real-Time Workshop builds applications from Simulink diagrams for prototyping, testing, and deploying real-time systems on a variety of target computing platforms, including Texas Instruments C6000 class DSP processors. Users of Real-Time Workshop can direct it to generate source code that accommodates the compilers, input and output devices, memory models, communication modes, and other characteristics that their applications may require.

Real-Time Workshop is an extension of capabilities of Simulink and Matlab that automatically generates, packages and compiles source code from Simulink models to create real-time software applications on a variety of systems. By providing a code generation environment for rapid prototyping and deployment, Real-Time Workshop is the foundation for production code generation capabilities. Along with other tools and components from The MathWorks, Real-Time Workshop provides Automatic

code generation tailored for a variety of target platforms A rapid and direct path from system design to implementation Seamless integration with MATLAB and Simulink.

6.4 Use of Real Time Workshop

Once the Simulink model has been simulated it is possible to do some settings in model configurations in order to instruct Matlab to start the DSP code building process. The resulting translation will transform the Simulink model into a Code Composer Studio C language project and then, controlling the DSP compiler, into a DSP binary code, which can be downloaded on the DSK board.

The main steps regarding the implementation of DSP software can be summarized as follows:

- *Installing development tools*
- *Test of Simulink to Code Composer Studio link*
- *Implementation of the reduced algorithm for LPC voice conversion: test with input audio signal from Matlab workspace*
- *Test on Signal-to-noise ratio*
- *Optimization strategy: use of embedded coder ERT*
- *Optimization of simulation parameters*
- *Profiling strategy by means of DSP/BIOS: percentage of CPU use and profiling of subsystems*
- *Loading optimization: USB emulator*
- *Complete voice transformation algorithms: critical subsystems to be reprogrammed (work in progress)*

The model for voice conversion based on LPC analysis and synthesis primarily developed in Matlab language as been ported to Simulink with reduced functionalities in order to develop in parallel the DSP application and to test the system.

The work of rapid prototyping to implement on a DSP hardware the voice conversion based on LPC analysis and synthesis algorithms have been carried out starting from the Simulink model. Real-Time Workshop (RTW) tool have been used to automatically convert the Simulink code to a C code compliant with the DSP target.

Next the Code Composer Studio (CSS) framework have been used to compile and upload the C code into the DSP target. To be successful this chain of rapid prototyping requires the tuning of different aspects regarding the Simulink model itself, the RTW (Real Time Workshop) and the CSS Code Composer Studio) parameters.

In this phase of the work, our strategy has been to concentrate the effort in two main aspects:

1. adjusting the model to be compliant to the DSP target
2. selecting RTW and CSS parameters which optimize the code production.

In order to be more flexible in the implementation of new model features we have decided not to do any manual change in the C code. This approach has required to gain a specific knowledge in the tuning of all aspects of the rapid prototyping chain.

Following these guide-lines a number of changes have been implemented in the original simulation model: use of the “Signal Processing Blockset” library instead of similar functionalities available in the generic Simulink blockset.

Further changes have been made to select blocksets parameters appropriate for the DSP logic.Tests have been made in specific aspects of the simulation in order to obtain a model generating a more efficient DSP code.

As first result of this in progress work, the reduced model for voice conversion based on LPC analysis and synthesis, named *fontanalpc_11_10_04.mdl,* has been successfully compiled and uploaded on the Texas Instruments C6711 DSP target.

The model fontanalpc_11_10_04.mdl (fig.10) requires, as input, a source and a target voice that have to be synchronized. For the test run we have used a short pre-recorded sequence of vowels that is upload in the DSP hardware together with the code of the model. The output of the model is connected with the DAC line out.

In order to produce a C code compliant with the DSP hardware a number of “Configuration parameters” of the The Real-Time Workshop tool have been customize:

- *Solver parameters*: the type of solver to be used to solve the currently selected model have been selected as *Fixed-step*; The fixed step size (fundamental sample time) used by the selected fixed-step solver have been set equal to 1/8000 seconds.
- *Optimization parameters*: default value have been taken.
- *Hardware Implementation parameter*s: the device type of hardware that will be used toimplement the production version of the system represented by the model have been set to TI C6000.
- *Real-Time Workshop parameters*: the system target file have set to “Embedded Target for TI C6000 DSP (GRT)”; the code generation target type, in the “TI C6000 target selection” sub panel, have been selected to C6711DSK; the overrun action, in the “TI C6000 runtime” sub-panel, have been selected to Notify and Continue. The other parameters have been left in the default state.

A first analysis of the simulation agreement with the DSP target have been made by using the *Model Advisor* [20] tool included in the Real-Time Workshop.

The Model Advisor is a software tool that comprehensively analyzes the Simulink model to help the programmer to appropriately configure Simulink and Real-Time Workshop. This tool analyzes model and block-level configuration and reports findings to the screen. The programmer controls the tool from a browser window, from which he can select from a set of checks on a model's current configuration. Model Advisor then generates a report describing both, good and bad conditions, providing suggestions for improvements in each area.

The Model Advisor analyzes the following set of features:

- *Identify unconnected lines, input ports, and output ports*
- *Check root model Import block specifications*
- *Check solver for code generation*
- *Identify questionable blocks within the specified system*
- *Check the hardware implementation*
- *Check optimization settings*
- *Identify questionable software environment specifications*
- *Identify questionable code instrumentation (data I/O)*
- *Check for parameter tunability information ignored by Simulink*

- *Check for implicit signal resolution*
- *Check for optimal bus virtually*
- *Identify blocks that generate expensive saturation and rounding code*
- *Check sample times and tasking mode*
- *Check for Mux blocks used as Bus Creator blocks*

Our main finding is that the Model Advisor is a necessary tool for a first step in the rapid prototyping but is not sufficient for a further optimization of the code generation.

Thus, using the Model Advisor the fontanalpc_11_10_04.mdl have been tuned and the result has been a model syntactically correct for a rapid prototyping. Furthermore, the fontanalpc_11_10_04.mdl model have been manually optimized and successfully compiled and uploaded on the DSP target.

7 Preliminary results

The reduced model (fig.10) has been successfully converted and loaded into the DSK boards. The signal at the DSK DAC output exactly reflects the simulation output.

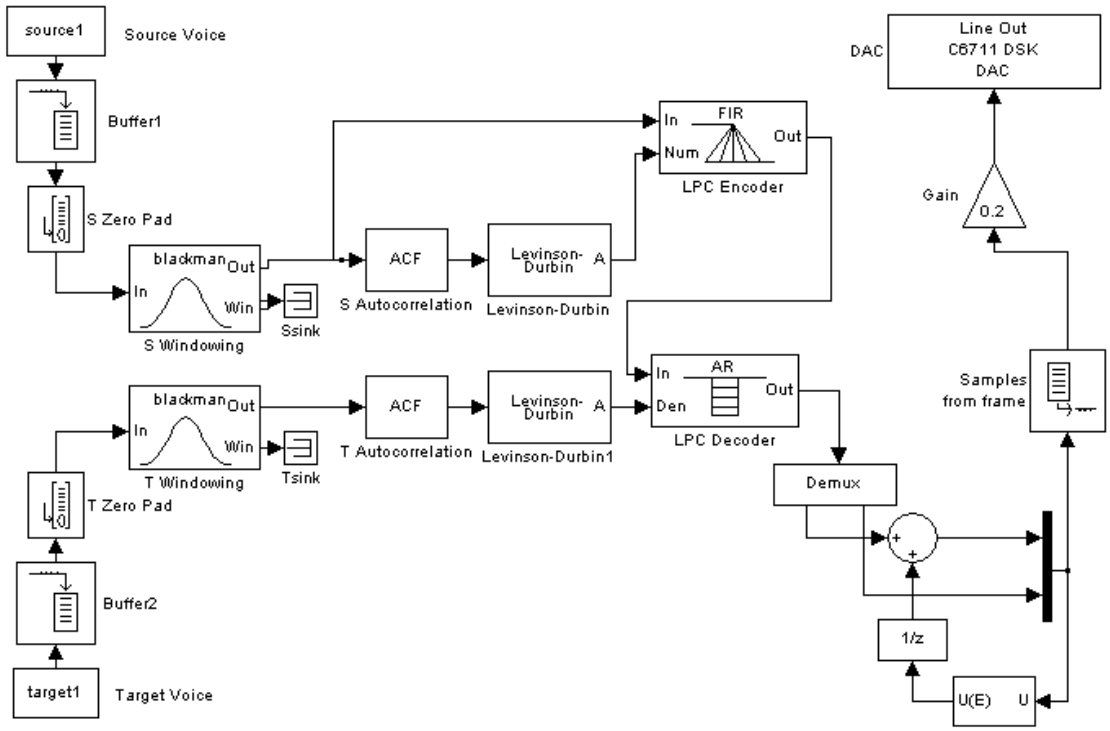

Fig. 10 **Reduced model of the voice conversion algorithm.**

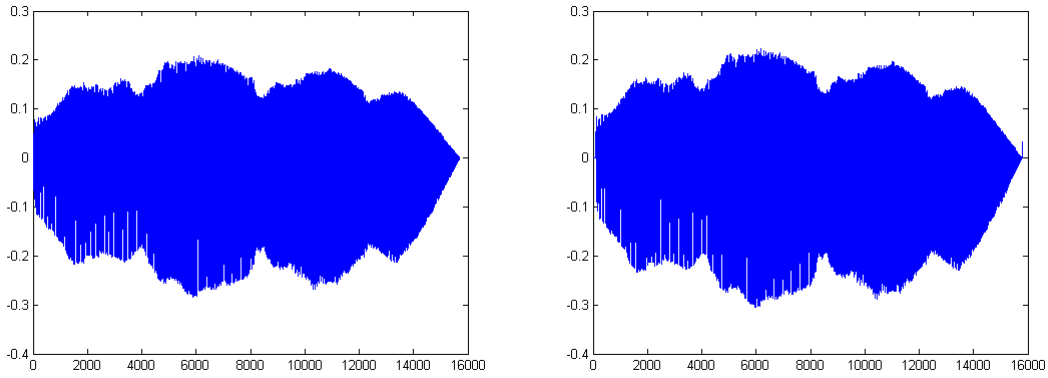

Fig. 11 Source Vs. Synthesized voice waveform

Fig.11 shows the comparison between a sampled voice, sent on both Source and Target algorithm inputs (left), and the resulting waveform obtained at output (right).
The shown output signal is obtained using the PC simulation: however the analog output signal of the board, beside some background noise, is exactly the same.

This model requires, as input, a source and a target voice that has to be synchronized. For the test run we have used a short preregistered sequence of vowels that is upload in the memory of the DSP motherboard. The output of the model is connected with the DAC line out.
By executing this model the DSP warns for a CPU overrunning, while the CPU load is about the 57%. By investigating this anomaly we have found the overrun was caused by the reading in the motherboard memory of the source and the target voice. In consequence of that result we have modified the Simulink model to read both the source and the target voice from the ADC line input. As previously described, the C6711 DSK motherboard has an input ADC and output DAC based on a mono 16 bit resolution and 8 KHz sampling rate Codec; the reading of both source and target by the same channel has no algorithmic meaning but is useful for performance investigation.
With this change the Simulink model run on the DSP without CPU overrun and with a CPU load of about the 35%. This result is important do to the fact that the voice conversion algorithm will read the source by the line in and will load a reduced number of parameters in the memory of the DSP motherboard.
Considering that our goal is to run an algorithm for voice conversion on board of the DSP and that for this algorithm the computational request will largely increase, we have investigate a number of strategies to gain a more performing code.
From one side we have investigated the fine tuning of a number of configuration parameters of Real-Time Workshop that can be implemented to increase the compliance and the performance of the DSP code. The result of this work has been a more stable DSP code.

Furthermore, in order to gain also a more efficient code, we have investigate the use of an additional Simulink library, the Real-Time Workshop Embedded Coder. The Real-Time Workshop Embedded Coder is intended to provide a production of code for the embedded system which is optimized for speed and memory usage. We have proved that the use of the Real-Time Workshop Embedded Coder library reduces of about 10% the CPU load. On the base of the previous results we have decided to use the Real-Time Workshop Embedded Coder library.

Nevertheless we experienced some problems during this testing phase. The required SW setup, that is Matlab/Simulink and Code Composer Studio running together on the same PC, is rather heavy and requires a quite powerful machine for avoiding unexpected system errors and crashes. We also experienced communication problems, because of the non perfect reliability of parallel port connection with the target DSK board.

For this reason we tried the algorithm also on another model of DSK board, based on TMS320C6713 DSP, newer and a bit faster. This board, still supported by real time workshop, uses a more reliable USB connection with the PC host.

The data sheet reports that the C6713 DSP, operating at 225 MHz, delivers up to 1350 MFLOPs whereas the C6711 DSP, operating at 150 MHz, delivers up to 600 MFLOPs.

8 Conclusions

The main goal of the project described has been the development of instruments and methods for reconstructing audio sequences starting from old, highly damaged, movie films. A solution based on LPC/VC algorithms plus GMM transformation has been proposed. The chosen solution provides successful approach to the physical modeling of the vocal tract, and physical meaningfulness and directivity of the control parameters. An improvement the re-synthesized voice based on non-linear techniques is currently under development: in this way the system becomes an efficient emulator of the perceptive function characterized by a low-dimensional set of parameters.

These algorithms have been firstly tested by Mathwork's Simulink environment. High quality voice reconstruction tests has been made with good results. According to one of the project's goal, we tested the possibility of a real-time implementation using a modern DSP platform provided by Rapid Prototyping tools. Considering the high complexity of the whole algorithm, together with some limits of the HW/SW development platform used, the real time implementation has been carried out only in a simplified form.

An interesting result of becoming familiar with rapid prototyping approach has been achieved: we found the used work flow very powerful and attractive for further investigations and improvements.

9 Acknoledgements

The authors wish to thank Massimiliano Dedola and Lucio Di Giovannantonio in supporting some parts of the HW/SW development.

10 Bibliography

1. RACINE-S EU Project IST 2001 -37117.

2. Bertini G., Di Giovannantonio L., Gonzales D., Grassi L., Magrini M., Dedola M.

"Sistema di sviluppo con DSP floating-point per il rapid prototyping". Nota interna ISTI-CNR, B4-14 Dec. 2004

3. E. Moulines and F. Charpentier, "Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones," *Speech Commun.*, vol. 9, pp. 453-467, 1990.

4. T. Dutoit and H. Leich, "MBR-PSOLA: Test-To-Speech synthesis based on an MBE re-synthesis of the segments database," *Speech Commun.*, vol. 13, pp. 435-440, 1993.

5. R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-34, no. 4, pp. 744-754, August 1986.

6. T. F. Quatieri and R. J. McAulay, "Shape invariant time-scale and pitch modification of speech," *IEEE Transactions on Signal Processing*, vol. 40, no. 3, pp. 497-510, March 1992.

7. Y. Stylianou, "Applying the harmonic plus noise model in concatenative speech synthesis," *IEEE Trans. on Speech and Audio Processing*, vol. 9, no. 1, pp. 21 -29, January 2001.

8. A. Syrdal, Y. Stylianou, L. Garisson, A. Conkie, and J. Schroeter, "TD-PSOLA versus Harmonic plus Noise Model in diphone based speech synthesis," *in Proceedings of IEEE ICASSP'98*, pp. 273-276, 1998.

9. A. Kain and M. W. Macon, "Spectral voice conversion for text-to-speech synthesis," *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 285-288, 1998.

10. A. Kain and M. Macon, "Personalizing a speech synthesizer by voice adaptation," *in Proc. of the 3rd ESCA/COCOSDA Int. Workshop in Speech Synthesis*, pp. 225-230, 1998.

11. A. Kain and M. W. Macon, "Design and evaluation of a voice conversion algorithm based on spectral envelope mapping and residual prediction," *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2001.

12. 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, March 1998.

13. N. Ivahashi and Y. Sagisaka, "Speech spectrum conversion based on speaker interpolation and multi-functional representation with weighting by radial basis function networks," *Speech Commun.*, vol. 16, pp. 139-151, 1995.

14. M. R. Schroeder, "Linear Predictive Coding of Speech: Review and Current

Directions," *IEEE Communication Magazine*, vol. 23, no. 8, pp. 54-61, 1985.

15.E. de Boer, Pitch of inharmonic signals, Nature, 178, pp. 535-536, (1956).

16. J. Cartwright, D.L. Gonzalez and O. Piro, "Nonlinear dynamics of the perceived pitch of complex sounds" Physical Review Letters, vol. 82, n. 26, pp 5389-5392 (1999).

17. J. Cartwright, D.L. Gonzalez and O. Piro, "Pitch perception: A dynamical-systems perspective" PNAS, vol. 98, n.9, pp. 4855-4859 (2001).

18.D.L. Gonzalez and O.A. Rosso, Qualitative modeling of complex biological systems, Proceedings of the Second Italian-Latinamerican Meeting of Applied Mathematics, pp. 132-135 Roma, (1997).

19.K.Konstantinides, "VLIW Architectures for Media Processing", IEEE Signal Processing Magazine, March 1998, pp.16-19

20. The Mathworks, MATLAB Documentation