Embedded Unit Selection Text-to-Speech Synthesis for Mobile Devices

Sotiris Karabetsos, *Member*, IEEE, Pirros Tsiakoulis, *Member*, IEEE, Aimilios Chalamandaris, and Spyros Raptis

Abstract — *Nowadays, unit selection based text-to-speech technology is the mainstream approach for near natural speech synthesis systems. However, this is achieved at the expense of raised requirements in terms of computational resources. This work describes design and implementation approaches for the efficient integration of this technology in computational environments with limited resources, such as mobile devices, with no considerable speech quality degradation. In particular, the issues of database reduction, acoustic inventory compression and runtime computational load minimization are mainly addressed in this paper. Both objective and subjective assessments confirm the effectiveness of these approaches in terms of constructing a general purpose embedded unit selection TTS system and reducing the computational requirements while maintaining high speech quality*[1].

Index Terms — Embedded Speech Synthesis, Unit Selection, Text-to-Speech, Mobile Devices, Mobile Phones.

I. INTRODUCTION

Text-to-Speech (TTS) technology deals with the production of synthetic voice output using textual information, thus serving as a more natural interface in human machine interaction. In order for TTS technology to be widely adopted, near-natural voice output quality has to be achieved. Over the last years, significant research progress in the field has contributed towards this goal. Nowadays, unit selection concatenative speech synthesis technology has become the dominant approach for building naturally sounding text-to-speech systems. This technique relies on runtime selection and compilation of speech units from a large speech database [1]. The speech database usually consists of a sufficient corpus of appropriately selected naturally spoken utterances, carefully annotated to the unit level. In most cases the speech units are phonemes or diphones. Each utterance comes from a text corpus designed to cover as many units as possible in different phonetic and prosodic contexts. The resulting repository of speech units may have little or great redundancy, on which speech variability and overall quality is significantly depended [1]. The drastic improvement in quality of synthetic speech, namely naturalness and intelligibility, over the years has led to the adoption of TTS as a mainstream technology. As a result, TTS technology is now employed in a wide range of applications, spanning from assistive technology and education, to telecommunications and entertainment [2]-[5]. However, this usually comes at the cost of large resource repositories and increased processing power, limiting the applications in desktop or server-based environments and therefore prohibiting its use on embedded or portable devices. Yet, the ever increasing demand for enhanced consumer applications and the extensive use of portable devices such as mobile phones or personal digital assistants (PDA) in everyday's life, have intensified the need to efficiently adapt TTS technology in environments with limited computational resources. For example, application areas such as assistive aids and tools, speech-to-speech translation, robotics, mobile phones, household devices, navigation and personal guidance gadgets, can largely benefit from the more natural and intuitive means of human computer interaction (HCI) offered by speech [6]-[10].

In order to address the challenge of developing a high quality TTS system for embedded devices, several approaches have been considered, regarding not only the adaptation of unit selection TTS but also the underlying technology itself. In reference to the technology, there are several different approaches for building low footprint TTS systems. Solutions include diphone-based TTS where the speech stimuli are comprised of only one instance of every diphone, and parametric TTS, such as Formant-based or HMM-based [1], [11]-[13]. Even though both diphone-based and formant-based TTS systems are well suited for low resource TTS, they suffer from degraded speech quality which is often not acceptable for mainstream consumer products [11]. On the other hand, recent results have shown that statistical parametric speech synthesis based on Hidden Markov Models (HMM-based TTS) can deliver high quality synthetic speech with reduced demands for computational resources, and hence can efficiently be adopted for portable devices [12].

Nevertheless, unit selection TTS is still the dominant approach for high quality speech synthesis, and therefore its efficient adaption to environments with reduced computational resources is of great interest so long as the speech quality is preserved. Recent research in this field has been mainly concentrated on optimizing several aspects of the speech synthesis procedure. These aspects include speech database reduction and compression for unlimited domain speech synthesis, speech signal parameterization and runtime synthesis optimization [11]-[16]. Approaches for limited

[1] This work was supported in part by E.U. and National funding.

S. Karabetsos, P. Tsiakoulis, A. Chalamandaris and S. Raptis are affiliated with the Institute for Language and Speech Processing (ILSP) / R.C. Athena, Department of Voice & Sound Technology, Artemidos 6 & Epidavrou, Marousi, GR 15125, Athens, Greece (e-mail: {sotoskar, ptsiak, achalam, spy}@ilsp.gr).

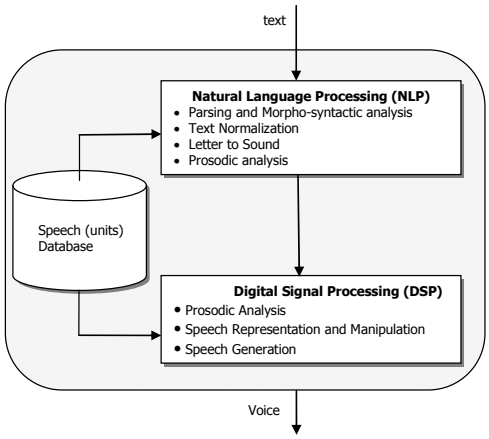

Fig. 1. General architectural diagram of a corpus-based TTS system.

domain speech synthesis (that is, domain-specific speech databases e.g., weather forecast) have also been proposed [17]. In summary, the most important issues of embedded unit selection TTS, relate to the objective of efficiently balancing the requirements regarding the computational load and the available resources together with delivering high quality speech.

In this paper, we describe design and implementation approaches for integrating generic domain unit selection TTS technology in environments with limited resources and computational capabilities such as mobile phones. More particularly, emphasis is given in three main issues. Firstly, we detail on the development of a specific methodology for the construction of a speech database for embedded devices, based on the existing databases utilized in the server-based or desktop-based versions of the corresponding TTS systems. The method relies on statistical analysis on the data derived from the unit selection stage on a large text corpus and employs, not only the selection frequencies, but also the unit selection scores of the units, leading to enhanced coverage and reduced redundancy. Secondly, we focus on the compression and coding of the speech units, aiming to efficient storage and retrieval, as well as to final signal quality during synthesis runtime. For this purpose, a code excited linear prediction (CELP) based approach is utilized and adapted to the particular needs of the TTS system. Finally, the last aspect we cover is the minimization of the computational requirements inherent to the unit selection module. The latter module performs a computationally demanding search to determine the optimal sequence among candidate speech units. To reduce the runtime computational load, we adopted a vector quantization (VQ) approach for the spectral join feature vectors of the same speech units and the offline computation of the corresponding distances. Evaluation results made clear that the aforementioned processes perform efficiently, leading successfully to a commercial TTS system for mobile phones of very high quality.

The rest of this paper is organized as follows. In section II, the unit selection concatenative speech synthesis technology is briefly reviewed and a description of the embedded TTS system architecture is given, highlighting its core modules. Section III provides details on the proposed design and implementation for database reduction and compression, as well as the minimization of the computational requirements related to the unit selection module. In section IV, both subjective and objective evaluation results are presented regarding the assessment of the followed techniques. Finally, a summary and some conclusive remarks are given in section V.

II. Embedded Text to Speech System Architecture

The general architectural diagram of a corpus-based TTS system is depicted in Fig. 1. There are two main components that comprise such a system, namely the Natural Language Processing (NLP) and the Digital Signal Processing (DSP). This diagram is valid for every data driven (that is, corpus-based) TTS system, regardless of the underlying technology (e.g., unit selection or parametric). The NLP component accounts for every aspect of the linguistic processing of the input text, whereas the DSP component accounts for speech signal manipulation and generation. For a unit selection TTS, besides the speech units (usually diphones) the speech database contains all the necessary data for the unit selection stage of the synthesis [1], [5].

In particular, the NLP component is mainly responsible for parsing, analyzing and transforming the input text into an intermediate symbolic format, appropriate to feed the DSP component. Furthermore, it provides all the essential information regarding prosody, that is, pitch contour, durations and intensity. It is usually composed of a text parser, a morpho-syntactic analyzer, a text normalizer, a letter-to-sound module and a prosody generator. All these sub-components are necessary for the disambiguation and proper expansion of all abbreviations and acronyms, for the correct word pronunciation, and also for the detection and application of the rich set of distinctive features of the speech signal, closely related to prosody.

The DSP component comprises of all the essential modules for the proper manipulation of the speech signal, that is, prosodic analysis and modification, speech signal representation and generation. Among various algorithms for speech manipulation, *Time Domain Pitch Synchronous Overlap Add* (TD-PSOLA), *Harmonic plus Noise (HNM)*, *Linear Prediction based* (LPC-based) and *Multiband Resynthesis Overlap Add* (MBROLA) are the ones that are mostly employed [1]. The DSP component also includes the unit selection module, which performs the selection of the speech units from the speech database using explicit matching criteria. More details about this module are given later in this section.

It becomes apparent that a full scale deployment of a unit selection TTS system is either infeasible or impractical in embedded environments. The system architecture that we

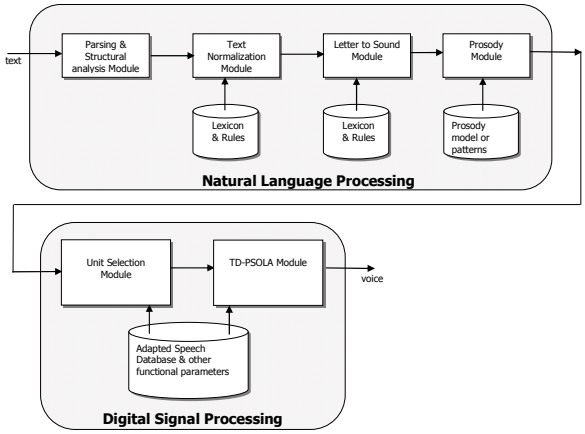

Fig. 2. System architecture for the embedded unit selection TTS system.

adopted for embedded unit selection speech synthesis is shown in Fig. 2. The core modules of our system are briefly described in the remainder of this section, while in section III, the adaptation techniques are explained.

A. NLP

As shown in Fig. 2, the input text is fed into the parsing module, where sentence boundaries are identified and extracted. This step is important since all remaining modules perform only sentence-level processing. The identified sentences are then fully expanded by the text normalization module. Besides numbers, abbreviations and acronyms, care has to be taken regarding the special characteristics of embedded environments (e.g., mobile phones). For such cases, particular care must be taken for the proper manipulation and expansion of special cases such as abbreviated menu options, stress disambiguation and alternative ways of text writing (e.g., multilingual, misspelled or transliterated such as in the case of *greeklish* [18]). To deal with these issues, the text normalization module relies on a rule-based approach combined with lexicon resources. A detailed presentation of these issues is beyond the scope of this paper and can be found at the given references [18], [19]. The letter-to-sound module transforms the expanded text in an intermediate symbolic form related to phonetic description. Most often, the module relies on a rule-based approach with limited computational requirements [19], complemented when necessary by exception dictionaries. The same approach is followed regarding prosody analysis, where pitch patterns are extracted by a large corpus analysis.

B. DSP

The DSP component comprises of the unit selection module and the signal manipulation module, in our case one based on TD-PSOLA. The speech database of the embedded TTS system uses a sampling frequency of 16 KHz. The database includes diphones as principal speech units, derived from the recordings of a Greek female professional speaker.

The unit selection module is considered to be one of the most important components in a corpus-based unit selection concatenative speech synthesis system. It provides a mechanism to automatically select the optimal sequence of database units that produce the final speech output, the quality of which depends on its efficiency. The criterion for optimizing is the minimization of a total cost function which is defined by two partial cost functions, namely the target cost and the concatenation cost function [1], [5].

The target cost function measures the similarity of an applicant unit with its predicted specifications (from NLP) and is defined as,

$$C^t(t_i,u_i)=\sum_{j=1}^{p} w_j^t \cdot C_j^t(t_i,u_i) \tag{1}$$

where, $u_1^n=\{u_1, u_2, \ldots, u_n\}$ are the candidate (sequence) units, $t_1^n=\{t_1, t_2, \ldots, t_n\}$ are the target (sequence) units, $C_j^t(t_i,u_i)$ is a partial target cost, p is the dimension of the target feature vector and w_j^t is a weighting factor for every partial target cost. The target feature vector typically employs target values for prosody and contextual features. The concatenation (or join) cost function accounts for the acoustic matching between pairs of consecutive candidate units and is defined as,

$$C^c(u_{i-1},u_i)=\sum_{j=1}^{q} w_j^c \cdot C_j^c(u_{i-1},u_i) \tag{2}$$

where, $C_j^c(u_{i-1},u_i)$ is a partial join cost , q is the dimension of the join feature vector and w_j^c is a weighting factor for every partial join cost. The feature vector typically consists of spectral similarity measures, pitch similarity measures, context similarity etc. Hence, the total cost is defined as,

$$C(t_1^n,u_1^n)=\sum_{i=1}^{n} W^t \cdot C^t(t_i,u_i)+\sum_{i=2}^{n} W^c \cdot C^c(u_{i-1},u_i) \tag{3}$$

or based on (1) and (2) it can be written as,

$$C(t_1^n,u_1^n)=\sum_{i=1}^{n} W^t \cdot \sum_{j=1}^{p} w_j^t \cdot C_j^t(t_i,u_i)+\sum_{i=2}^{n} W^c \cdot \sum_{j=1}^{q} w_j^c \cdot C_j^c(u_{i-1},u_i) \tag{4}$$

where, W^t and W^c are the weights that denote the significance of the target cost and the join cost, respectively. The goal of the unit selection module is to perform a (computationally demanding) search so as to find the speech unit sequence which minimizes the total cost, hence to specify,

$$\hat{u}_1^n=\min_{u_1 \ldots u_n} C(t_1^n,u_1^n) \tag{5}$$

The selection of the optimal speech unit sequence incorporates a thorough search (usually a Viterbi search) which involves comparisons and calculations of similarity measures between all available units, often employing heuristics to guide and/or limit the search [1], [5].

III. Adaptation Techniques for Embedded Devices

For the efficient adaptation and integration of unit selection TTS technology in embedded environments, a balance must be stroke between the conflicting demands of minimizing the computational load while preserving a high-quality speech output. In most of the cases, computational load increases in

proportion to the speech database size. As already mentioned, desktop- or server-based TTS systems involve large speech databases. To avoid building a speech database from scratch, effective scaling-down can be achieved by employing speech database reduction techniques. Furthermore, the resulting speech database should be compressed and encoded in a way suitable for synthesis runtime. Moreover, the computational requirements of the unit selection module need to be reduced. In the following sections, techniques to cope with these problems are described.

A. Speech Database Reduction

The method relies on statistical data produced by the full scale unit selection process on a large text corpus, initially proposed in [16]. It utilizes the selection frequency, as well as the actual score of each speech unit (diphone). As outlined in [15] the strategies usually fall in two categories: the top-down and the bottom-up approaches. According to the top-down approach, the unit repository is investigated for the reduction process and a clustering process is performed, based on prosodic and phonetic properties. By doing so, the search space of the unit selection algorithm is reduced as each target unit is searched within the corresponding cluster. On the other hand, the bottom-up approach is purely a data driven technique since it focuses on the statistical behaviour of the unit selection algorithm. The output of the unit selection stage is statistically analysed in order to reduce the unit repository. The statistical data is collected from the synthesis of a large text corpus, where the selection frequency of each unit is usually calculated and is used in the reduction process. For example, in [15] the removal of the less frequent units is proposed. A possible weakness of using only the selection frequency is that it does not help avoid redundancy. For example two very similar units that are alternatively selected equally often by the algorithm, will both be included in the reduced database.

The method proposed in this work falls into the bottom-up category, and overcomes the aforementioned problem, by employing a technique motivated by the clustering idea of the top-down approach. The truncation process is based on the selection frequency as well as the actual score of each unit during the unit selection process. More specifically, the difference of the scores between two instances of the same diphone is used as a similarity metric between them.

The main idea of the proposed method is to preserve the units that are most frequently used by the unit selection algorithm, and at the same time, avoid keeping similar units. The unit selection algorithm is run upon a sufficiently large text corpus, and statistical data is collected for the truncation process. Based on the unit selection algorithm described in section II, we define a score function for each unit in a synthesized utterance as the combined local target and join cost,

$$S(u_j^n) = C^t(t_n, u_j^n) + \min_k (C^c(u_j^n, u_k^{n-1})) \quad (6)$$

where, u_j^n denotes the *j-th* instance of the *n-th* diphone and t_n is the target diphone. The second term of the score function is a look-behind cost function and it expresses the best join cost of the unit u_j^n from all the instances of the previous diphone u^{n-1} in the utterance under consideration. This is used because a forward Viterbi search is used to find the best path. Alternatively a look-ahead cost function or both could be used, with the main principle of the method remaining the same. As far as the algorithmic point of view is concerned, if two instances of the same diphone score the same (or similar) in a given utterance, they are seen as similar ones, regardless of their objective similarity. This also derives from (4) since C^t and C^c are summed to find the best path. Thus, we can use the difference of scores, averaged over the whole corpus, as a similarity metric between instances of the same diphone. For all the utterances processed by the algorithm the following statistical data are collected: *i)* the selection frequency f_j^n for each u_j^n, namely the total number of times the unit was selected and, *ii)* the mean score difference $D_{j,k}^n = \overline{|S(u_j^n) - S(u_k^n)|}$ for all pairs of units of the same diphone (with scores referring each time to same utterance).

The reduction method relies on both the aforementioned quantities in order to select the appropriate instances of a specific diphone. Let K be the number of instances of a diphone in the available (large) database and $M < K$ the desired number of instances in the small database, then a greedy algorithm is used to select M units as shown in table I:

TABLE I
ALGORITHM FOR DATA BASE REDUCTION

1. Initialize $F = [f_1^n, f_2^n \ldots f_K^n]$
2. Select $m = \arg\max_n F[n]$
3. Update $F[n] = F[n] \cdot D_{n,m}^i$ for $n = 1 \ldots K$
4. If #(*selected*) < M goto step 2

We define F as the fitness vector of the instance units, initialized with the selection frequencies. Next, we iteratively select the unit with the best fitness. The most important step of the algorithm is step 3, in which we update the fitness vector to avoid selecting similar units. This is not done explicitly but, motivated by the idea of *fitness sharing* used in genetic algorithms. The fitness vector is updated after each selection. The fitness of each unit is multiplied with its mean score difference with the last selected instance. By doing so, the fitness of the similar units to the selected ones deteriorates, while different ones become more fit. As a side-effect, $F[m]$ becomes zero, thus already selected units cannot be reselected. The target value M for the number of units in the reduced database is determined by the desired reduction rate. The coverage meeting criterion (e.g. as proposed in [15]) cannot be used since frequent units may be discarded by the method as well as a specific coverage does not guarantee the reduction

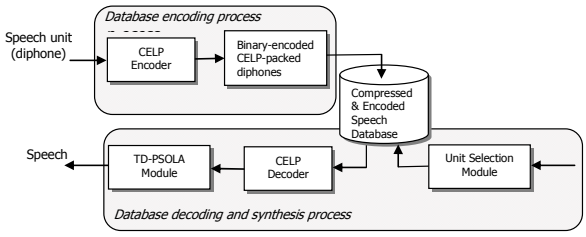

Fig. 3. Speech database compression and encoding by CELP adaptation.

rate in all diphones. An explicit function is utilized to determine the number of units:

$$M = min(M_{max}, max(M_{min}, log_b(K))) \quad (7)$$

where, the parameters M_{max} and M_{min} ($M_{max} > M_{min}$) explicitly define the maximum and minimum number of instance units per diphone, while parameter b determines a logarithmic reduction rate distribution among diphones. Evaluation results on this technique are presented in section IV.

B. Speech Database Compression and Coding

Speech database compression is considered as a vital problem in embedded unit selection speech synthesis since it facilitates for better and efficient adaptation of the technology in this domain [11]. The problem is not different than conventional speech coding although there are issues that are specific to TTS technology. The compression technique should not only ensure compression efficiency but also avoid introducing perceived signal degradations. Furthermore, it should provide random access capability and fast decoding. On the other hand, encoding complexity is not an issue since it is performed offline.

In this work, we have adopted Code Excited Linear Prediction (CELP) as the speech database compression technique in view of the fact that it is a well established and widely deployed coding scheme, capable of producing adequate speech quality [20]. The adaptation of CELP for the purpose of embedded unit selection TTS is depicted in Fig. 3. To cope with random access capability, every speech unit (diphone) is separately compressed and encoded. Hence, the speech database consists of CELP parameters representing diphones that are binary encoded for effective database organization. It is important to note that in this approach, neither the time limits nor the pitch marks of the diphones are affected. Also, no perceived spectral degradation occurs. Furthermore, a scalable bit allocation scheme is used for obtaining different compression ratios. In practice, informal listening tests have shown that compression ratios between 7 and 10 could be used.

At the synthesis stage, only the selected (best path) diphones are decompressed for TD-PSOLA, thus eliminating any overhead. The CELP decoder is implemented using fixed point arithmetic for performance optimization.

C. Reduction of the Computational Requirements of the Unit Selection module

One of the most demanding tasks during synthesis runtime is that of the unit selection. The unit selection process involves dynamically searching and deciding on the "optimal" unit sequence over a lattice of available units. The performance of the unit selection algorithm is vital since it heavily determines the response time of the system. Today's speech databases with sizes ranging from a few MB to several GB, and incorporating hundreds of instances per speech unit, pose increased demands on CPU power. In large scale systems, such as in desktop- or server-based TTS, this is compensated, without loss in quality, by the available processing and storage power complemented by both heuristics (e.g. pruning) and clustering over similar units techniques [21][22]. However, in the case of embedded TtS, these techniques are not appropriate, since they rely on the plurality of remaining units. The latter assumption is not applicable for embedded unit selection where the databases used are already reduced and, therefore, the search space has already been sufficiently limited.

In this work, a vector quantization (clustering) approach is adopted in order to achieve lower computational and storage costs, for the purpose of spectral join cost calculation, since it is the most expensive task in the unit selection process. The approach is based on a within-type clustering of the spectral join feature vectors of the speech units (e.g. the clustering of the feature vectors of the same phoneme) and the offline computation of distances between the centers of each cluster. The approach is motivated by the idea presented in [22], although it differs significantly since it maintains the available search space. A similar approach has been mentioned in [23], but deals only with the case of a large scale TTS system and does not put focus on the particular characteristics for the deployment in embedded devices.

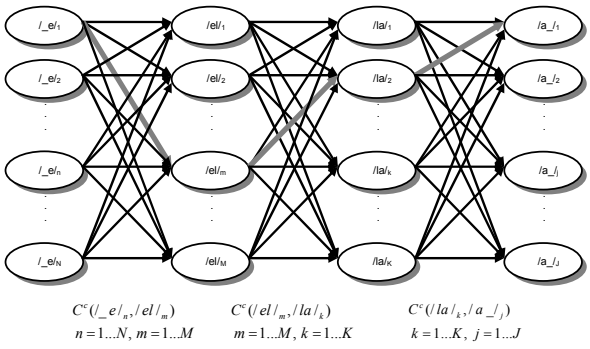

Fig. 4. The unit selection process. The best sequence path is based on the cumulative total score. An example of the best path is depicted with the solid grey line.

An example of the unit selection algorithm is illustrated in Fig. 4, where the synthesis of the utterance "ela" is depicted which is the Greek word for "come". We consider the use of diphones so the speech units are {/_e/,/el/,/la/,/a_/} each having a total of N, M, K and J number of instances in the

speech database respectively. Therefore, the corresponding lattice involves the computation of $N{\cdot}M+M{\cdot}K+K{\cdot}J$ concatenation cost, termed as C^c. It is important to notice that, among these metrics, the spectral join cost is one of the most predominant factors for high computational load and slow response time at synthesis runtime since unit selection is performed per utterance. It employs the retrieval of spectral feature vectors for each unit and the calculation of a distance which serves as the spectral join cost.

Specifically, if we let P = {*d*} be the phoneme set having |P| = N elements and D = {*pq*: *p*, *q* ∈ P and *pq* is valid} be the diphone set having |D| = M elements then the following statement is true, $N^2 \geq |D| = M \sim O(N^2)$. Furthermore, we consider the unit selection speech database as a repository of instances of speech units (diphones) that forms the set R = { u_k^{pq} : the k_{th} instance of $pq \in D$}. If we let K be the mean number of instances per speech diphone then, a specific phoneme may form N diphones as the left phoneme of a diphone and N diphones as the right phoneme of a diphone, each of them having K instances respectively. For example, the phoneme /a/ forms the diphone sets /aX/ and /Xa/ where X ∈ P (e.g. can be any of N phonemes). Since diphones for the type /Xa/ concatenate only with diphones of /aX/ and since there are K instances for each of them, the possible joins for a phoneme (in the diphone set) is in the order of N^2K^2. Thus, since the phoneme set has N elements, the order of (total) possible joins in R is $O(N^3K^2)$. Moreover, for each $u \in R$ we need two pairs of (concatenation) feature vectors, that is, one for the left phoneme (left join $v_L(u)$) and one for the right phoneme (right join $v_R(u)$) and a distance measure, $d(v_R(u_i^{aq}), v_L(u_j^{qb}))$ or for simplicity $d(v_R(u_i), v_L(u_j))$, between two units to be used as the join cost. Therefore, the options for storage and computation are, either to store the feature vectors $v_L(u)$, $v_R(u)$ for every $u \in R$ and evaluate the distance *d* at runtime, or compute every distance offline and store them (in tabular format) for every possible join in R. The former entails a high runtime computational cost while the latter a high storage cost.

In order to reduce the runtime computational and storage cost, a within-type clustering of spectral join feature vectors of speech units is proposed, as well as the offline computation and storage of distances between the centres of each cluster. More specifically, an offline clustering of all spectral feature vectors of the same phoneme is performed, followed by the computation of the distances between the centres (or representatives) of each cluster. Thus, the cluster distances are pre-computed and are used instead of the true distances for joining segments. This technique offers a low runtime computational and storage cost since it reduces the required number of concatenation costs calculations. However, this is achieved at the expense of possible degradation of the resolution of the spectral join cost which might affect the synthetic speech quality. Experimental evaluation shows that no significant degradation in quality is observed. The algorithmic description of the technique is illustrated in table II.

As a result, if C is the cluster size per phoneme and the concatenation occurs per diphone, the number of total possible cluster joins is in the order of $O(NC^2)$ which is $O(NC^2) << O(N^3K^2)$ since C can be chosen to be adequately small. Additionally, it is important to notice that the number of possible joins per diphone is C^2 for the case of clustering and $N{\cdot}K^2$ without clustering. Hence, the statement $C^2 < N{\cdot}K^2$ is true even for the case of embedded devices as long as the number of instances per diphone is sufficient and again if C is adequately small. For example, the developed Text-to-Speech system for mobile phones utilizes a phoneme set for the Greek language that has N = 34 elements and the number of instances per diphone is at least 10. Hence, if the cluster size is C = 32 the above criterion is met.

TABLE II
ALGORITHM FOR CLUSTERING SPECTRAL JOIN COSTS (OFFLINE)

1. $\forall p \in P$ do steps 2 to 5
2. Find all instances of speech units that have *p* as left or right phoneme i.e., find $R_{left}^p = \{ u^{lr} : u \in R \text{ and } p=l \}$ and $R_{right}^p = \{ u^{lr} : u \in R \text{ and } p=r \}$
3. Perform clustering of $\{v_L(u): u \in R_{left}^p \} \cup \{ v_R(u): u \in R_{right}^p \}$ in *C* clusters with centres c_i, $i = 1 \ldots C$
4. Compute $M^p(i, j) = d(c_i, c_j)$, $i, j = 1 \ldots C$
5. Store $M^p(i, j)$ and the two cluster indexes per unit instance

At synthesis runtime the distance between pairs of diphones is retrieved and calculated as $M(a, b)$ instead of $d(v_R(u_i^{yp}), v_L(u_j^{px}))$ where, a,b are the corresponding cluster indexes that each phoneme of every diphone belongs to. The method does not reduce the available search space since it is clear that no clustering on the speech units themselves can be performed since the speech database is already reduced. Instead, the search space is kept the same while clustering is carried out for the features that constitute the spectral join cost. While this may lead to resolution degradation, it is assumed that since within-cluster costs have small differences between them together with implicit compensation due to other sub-costs, the reduction of the cost resolution can be tolerated.

IV. EVALUATION AND RESULTS

The techniques addressed in this work are assessed using both objective and subjective criteria. For subjective evaluation, the most common approach in assessing the quality of TTS systems is through listening tests where a group of people is asked to express their opinion regarding the TTS quality namely, naturalness and intelligibility. The results, usually expressed in terms of mean opinion scores (MOS), reflect rather accurately the perceived quality of a

TTS system [1], [5].

The experiments were carried out on a database of a Greek female speaker, which consists of a total of 1291 annotated utterances from a phonetically balanced corpus of modern Greek language. The resulting complete database has a total of 1098 unique diphones and contains about 115K instances. The final (embedded) database has approximately 11K diphones. The total resources are approximately 6MB and the memory footprint of the TTS is less than 2MB. There are no separate evaluation results for CELP encoding and decoding since this process is implicit in the following evaluation experiments. Furthermore, the mobile phone utilized in the experimental evaluation had a CPU of 220MHz.

A. Speech Database Reduction Evaluation

After benchmarking with various target embedded devices, we reached to the conclusion that reasonably high reduction rates, up to 95%, are both possible and necessary for the TTS system to run efficiently. At such high reduction rates, a degradation of output speech quality is almost inevitable, especially as far as variability in the speech is concerned. A large text corpus of no specific domain was collected for testing purposes. Hence, a total of about 12.5K sentences covering about 1.5M diphone instances were utilized. A 95% segment of the corpus was used to collect statistical data from the unit selection synthesis algorithm, and the rest was used for the objective evaluation process. The most obvious method for comparison is the "*select most frequent units*" method [15]. In order to have meaningful results we use the same number of units per diphone M across methods. Hereafter we refer to our method as P_F and to the most frequent selection as P_S. As shown in [16], both methods fully overlap for extreme reduction cases. For the evaluation of the database reduction technique, objective metrics derived from statistical parameters describing the behavior of the unit selection algorithm, are utilized. The commonly used statistics are, the mean values of target, join and total costs over the best path units. In addition, another set of objective metrics, also derived from the statistics of the unit selection algorithm, are introduced. In particular, the maximum target, join and total cost is considered. By taking into account the maximum cost per utterance, we try to identify glitches in the synthetic speech, since places of high cost are potential prosodic, spectral or other types of discontinuities. Such cases are usually avoided with the use of a large database, but this may be inevitable at high reduction rates. All the above statistical metrics are calculated per utterance and averaged over the whole test corpus. The comparison results of the objective evaluation of P_F and P_S are illustrated in Fig. 5. As a reference point, the corresponding measures for the complete database system are $\{total, join, target\}_{mean} = \{0.15, 0.07, 0.07\}$ and $\{total, join, target\}_{max} = \{0.50, 0.27, 0.34\}$. Although P_S performs slightly better in terms of mean costs, P_F has a far lower average maximum cost per utterance, which becomes more pronounced as the reduction rate increases. This behavior indicates two main presumptions. The P_S method produces databases that result in synthetic utterances with good scores if averaged, but also having units with poor scores. On the other hand, P_F produced databases resulting in utterances with far better target cost at the cost of a slightly higher join cost.

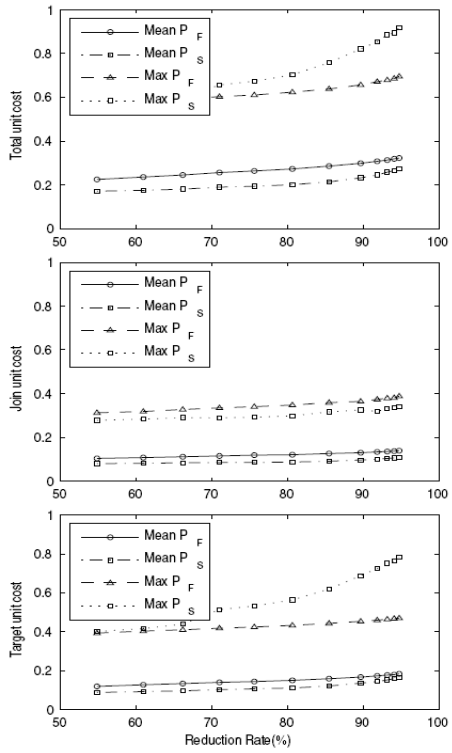

Fig. 5. Comparative objective evaluation between P_F and P_S. Top: the averaged mean (solid) and max (dashed) total cost per utterance for P_F and for P_S with dash-dotted and dotted lines respectively. Middle, bottom: the metrics for the join and target cost respectively are depicted with same notation.

TABLE III
MOS RESULTS OF THE DATABASE REDUCTION METHOD

	MOS	Mean Costs	Max Costs
P_F	4.01	0.32, 0.14, 0.18	0.61, 0.34, 0.40
P_S	3.92	0.27, 0.11, 0.16	0.84, 0.30, 0.71

In order to subjectively assess the method, listening tests with 35 short sentences (2 to 16 words long) selected from the test corpus were conducted. The sentences were synthesized with databases produced by P_F and P_S with a reduction rate of 93%. A group of 15 listeners, speech experts and listeners with no experience in synthetic speech were asked to evaluate each pair of sentences, presented in a shuffled order each time. The results are summarized in table III, where the mean opinion score (MOS) is shown together with the objective metrics (total, join, target costs). The results show that P_F produces better synthetic speech than P_S. Also, there is an agreement of the MOS values and the averaged maximum total cost per utterance. This seems to verify the initial hypothesis that P_S could result in redundant units in terms of

target features, by selecting more similar units and leaving at the same time no room for other units to cover other less frequent but equally important cases met in general purpose TTS systems.

B. Clustered Spectral Join Cost Evaluation

The acoustic representation that is used is Mel-Frequency Cepstral Coefficients (MFCC) and the Euclidean distance between MFCC vectors is used as a spectral join cost. Thus, the feature vectors to be clustered are the MFCC vectors for every phoneme of every diphone. For all the experiments in this work, the number of clusters per phoneme is set to C = 32 and clustering was performed using the *k-means* algorithm utilizing the Euclidean distance measure as a classification metric among the feature vectors.

In order to evaluate the performance of the proposed technique a comparison between two versions of the unit selection algorithm namely, with (C_{US} – clustered join cost unit selection) and without (F_{US} – full unit selection) clustering, has been implemented for spectral join cost calculation. A total of 52 sentences were synthesized using both versions and the averaged times concerning the benchmarks of the processing time of the unit selection module were measured. The results are summarized in table IV. Obviously, a significant reduction in the computational load is observed since the proposed technique results in the reduction of the computational time, for the unit selection module on average by a factor of more than three and improves the overall performance of the TTS system on average by a factor of 29% compared to the F_{US} version. Additionally, the C_{US} version accounts only for the 13.1% of the total processing time. Moreover, a 2.4 real time factor is achieved, on the specific mobile phone. Consequently, the response time of the TTS, which heavily depends on the unit selection module, is greatly reduced achieving a mean value of approximately 0.25sec.

TABLE IV
BENCHMARKS ON THE PROCESSING TIME OF CUS

Unit selection speed improvement	> 3.5 times
Total speed improvement	> 29%
Mean response time	0.25sec
Real time factor	> 2.4
Percentage of total TTS time	**F_{US}**: 32.5%
	C_{US}: 13.1%

To assess the effect of the proposed approach in the overall speech quality, we conducted a small scale acoustical experiment. A total of 52 short sentences, having 4 to 16 words, were synthesized (on a mobile phone) using both F_{US} and C_{US}. The sentences were no-domain specific and were not included in the speech database. A group of 15 listeners, comprised by both speech and non-speech experts, were asked to express their opinion for each sentence in terms of overall quality. Each sentence was presented in pairs (F_{US} and C_{US} version) and the subjects could listen to each sentence more than once. The order of each pair was random. The results are summarized in table V. The results depicts that the proposed technique performs slightly better, as far as overall quality is concerned, than its full version counterpart. However, the standard deviation shows that both versions can be considered equivalent. The main conclusion is that C_{US} results in a synthetic speech quality that is practically indistinguishable compared to the F_{US} version. On the other hand, the gain in computational time is significant. Additionally, the cost resolution degradation is well balanced since the clustering approach does not reduce the original search space, therefore any possible degradation is compensated by the target cost measures or other sub-costs involved in the concatenation cost calculation. This is in accordance with the experimental findings. Moreover, it is important to notice that the number of clusters causes a trade-off between processing time, storage and degradation in cost resolution. Indeed, as the number of clusters per phoneme increases the storage also increases. On the other hand, a small number of clusters cause a large number of phonemes to be represented by a single feature vector in the join cost calculation which does not account for properly scoring of acoustical dissimilarities and would lead to quality degradation.

TABLE V
SUBJECTIVE EVALUATION OF THE C_{US} TECHNIQUE

	MOS	**Standard Deviation**
F_{US}	3.98	0.45
C_{US}	4.01	0.39

C. Overall Results

Table VI summarizes computational benchmarks regarding the embedded unit selection TTS system.

TABLE VI
BENCHMARKS OF THE TTS SYSTEM

Module	**Computation Percentage at synthesis runtime**
NLP	3%
CELP decoding	69%
Unit Selection	13%
TD-PSOLA	15%
General characteristics	
Database size (scalable)	4-8MB
Real Time	≥ 2,5
Response Time	0.25sec

**Response time depends mostly on the unit selection module, followed by CELP decoding and TD-PSOLA.*

The results are depicting that the TTS system is capable of real time operation with low response time and is sufficiently scaled for embedded environments.

V. CONCLUSIONS

In this paper, we have described the system architecture of a general purpose embedded unit selection TTS system and we have presented efficient techniques that successfully address the challenging problems arising in embedded environments, such as database reduction, database compression, and runtime load minimization. In particular, we have presented an algorithm which leads to small footprint

speech databases with increased diversity and reduced redundancy. Sufficient compression ratios were achieved by appropriately adapting CELP to the synthesis process. Finally, a vector quantization approach was derived for the spectral join cost calculation that significantly reduces the computational requirements of the unit selection module. Evaluation results provide clear evidence of substantial improvement in the computational resources exploitation while preserving the overall speech quality in terms of naturalness and intelligibility. All the concepts and approaches proposed in this paper have been employed in the development of a top-quality embedded unit selection TTS system for the Greek language. The system has been successfully adopted as part of a screen-reader solution for mobile phones.

ACKNOWLEDGMENT

The authors would like to thank all the persons involved in the listening tests or contributed to this work.

REFERENCES

[1] T. Dutoit, "Corpus-based Speech Synthesis*," Springer Handbook of Speech Processing,* J. Benesty, M. M. Sondhi, Y. Huang (eds), Part D, Chapter 21, pp. 437-455, Springer, 2008.

[2] G. Bailly, W.N. Campbell, and B. Mobius, "ISCA Special Session: hot topics in speech synthesis", *Proc. Eurospeech 2003,* pp. 37-40, Geneva, 2003.

[3] B. Duggan and M. Deegan, "Considerations in the usage of text to speech (tts) in the creation of natural sounding voice enabled web systems", *In Proc. of the 1st international symposium on Information and communication technologies (ISICT '03:)*, pp. 433–438, Trinity College Dublin, 2003.

[4] N. Campbell, "Developments in Corpus-Based Speech Synthesis: Approaching Natural Conversational Speech," *IEICE trans. Inf. & Syst.*, vol. E88-D, no. 3, pp.376-383, 2005.

[5] Douglas O'Shaughnessy, "Modern Methods of Speech Synthesis," *IEEE Circuits and Systems Magazine,* Third Quarter 2007, pp. 6-23, 2007.

[6] J.-P. Peters, C. Thillou, and S. Ferreira, "Embedded Reading Device for Blind People: a User-Centred Design," *Proc. of 33rd Applied Imagery Pattern Recognition Workshop (AIRP'04)*, 2004.

[7] T. Schultz, A. W. Black, S. Vogel, and M. Woszczyna, "Flexible Speech Translation Systems," *IEEE trans. on Audio, Speech and Language Processing,* vol. 14, no. 2, pp. 403-411, 2006.

[8] S. Tomko, T. K. Harris, A. Toth, J. Sanders, A. Rudnicky and R. Rosenfeld, "Toward Efficient Human Machine Speech Communication: The Speech Graffiti Project," *ACM Transactions on Speech and Language Processing*, vol. 2, no. 1, Article 2, pp. 1-27, 2005.

[9] R. K. Moore, "PRESENCE: A human-inspired architecture for speech-based human-machine interaction," *IEEE Trans. Computers*, 56, pp. 1176-1188, 2007.

[10] L. Mohasi and D. Mashao, "Text-to-Speech Technology in Human-Computer Interaction", *5th Conference on Human Computer Interaction in Southern Africa, South Africa (CHISA 2006, ACM SIGHI)*, pp. 79-84, 2006.

[11] M. Schnell, O. Jokisch, R. Hoffmann, and M. Kustner, "Text-to-speech for low-resource systems," *IEEE Workshop Multimedia Signal Processing (MMSP)*, St. Thomas, pp. 259-262, 2002.

[12] S.-J. Kim, J.-J. Kim and M.-S. Hahn, "HMM-based Korean speech synthesis system for hand-held devices," *IEEE Trans. Consumer Electronics*, vol. 52, no. 4, pp. 1384-1390, 2006.

[13] N. Nukaga, R. Kamoshida, K. Nagamatsu, and Y. Kitahara, "Scalable implementation of unit selection based text-to-speech system for embedded solutions," *Proc. of IEEE ICASSP 2006*, pp. 849-852, Toulouse, 2006.

[14] D. Chazan, R. Hoory, Z. Kons, D. Silberstein, and A. Sorin, "Reducing the footprint of the IBM trainable speech synthesis system," *in Proc. ICSLP 2002*, Denver, CO, pp. 2381–2384. 2002.

[15] P. Rutten, M. P. Aylett, J. Fackrell, and P. Taylor, "A statistically motivated database pruning technique for unit selection synthesis," *in Proc. ICSLP 2002*, Denver, Colorado, USA, pp. 125–128, 2002.

[16] P. Tsiakoulis, A. Chalamandaris, S. Karabetsos and S. Raptis, "A Statistical Method for Database Reduction for Embedded Unit Selection Speech Synthesis," *in IEEE ICASSP 2008*, pp. 4601-4604, 2008.

[17] Y. Ishikawa, Y. Kisuki, T. Sakamoto, and T. Hase, "Speech Synthesis Method based on Application-Specific Synthesis Units and its Implementation on a 32-bit Microprocessor," *IEEE Trans. on Consumer Electronics*, vol. 45, no. 3, pp. 980-985, 1999.

[18] A. Chalamandaris, A. Protopapas, P. Tsiakoulis, and S. Raptis. "All Greek to me! An automatic Greeklish to Greek transliteration system." *5th International Conference on Language Resources and Evaluation (LREC 2006)*. Genoa, Italy, pp. 1226–1229, 2006.

[19] A. Chalamandaris, S. Raptis, and P. Tsiakoulis, "Rule-based grapheme-to-phoneme method for the Greek," *in Interspeech 2005*, pp. 2937-2940, 2005.

[20] Chu, Wai C. "*Speech coding algorithms: Foundation and evolution of standardized coders,*" John Wiley & Sons, 2003.

[21] Beutnagel, M., Mohri, R., and Riley, M., "Rapid unit selection from a large speech corpus for concatenative speech synthesis," Proc. Eurospeech 99, Budapest, 1999.

[22] Black, A., and Taylor, P., "Automatically clustering similar units for unit selection in speech synthesis," Proc. of Eurospeech 97, vol. 2, pp. 601-604, Greece, 1997.

[23] Coorman, G., Fackrell, J., Rutten, P., and Coile, B. V., "Segment selection in the LH realspeak laboratory TTS system," Proc. of the International Conference on Spoken Language Processing (ICSLP 2000), vol. 2, pp. 395-398, 2000.

Sotiris Karabetsos received the M. Eng. degree in Electrical Engineering and Computer Science from the National Technical University of Athens (NTUA), in 2004 and the M.S. degree in Data Communications from Brunel University of London, in 2003. He has also received the BS degree in Electronic Engineering from the Technological and Educational Institution of Athens (TEI of Athens), in 1999. He is currently working towards the Ph.D. degree in Speech Synthesis at NTUA. From 2003, he is with the Institute for Language and Speech Processing (ILSP). He is also with the Technological and Educational Institution of Athens (TEI of Athens), Department of Electronics. His research interests are speech synthesis, signal processing, and telecommunications. He is a member of IEEE.

Pirros Tsiakoulis received his M. Eng. degree in Electrical Engineering and Computer Science in 2003 from the National Technical University of Athens (NTUA), Athens, Greece. He is currently working towards the Ph.D. degree in Speech Synthesis at the National Technical University of Athens (NTUA). In 2000, he joined the Institute for Language and Speech Processing (ILSP). His research interests include NLP, speech synthesis and speech processing. He is a member of IEEE.

Aimilios Chalamandaris received his M.Eng. degree in Electrical Engineering and Computer Science from the National Technical University of Athens in 2000, and his M. Eng in Telecoms and Signal Processing from Imperial College in 2001. He is a PhD student at NTUA, and works at the Institute for Language and Speech Processing (ILSP), doing research on speech and signal processing. His research interests are NLP, speech synthesis, speech recognition, and signal processing.

Spyros Raptis received his M. Eng. degree in Electrical Engineering and Computer Science in 1994 and his PhD in hybrid computational intelligence for optimization, modeling and decision making in 2001, both from the National Technical University of Athens, Greece. He has been a lecturer at graduate and post-graduate level and has participated in a number of National and European RTD projects on speech technology, computational intelligence, robotics, and multimedia educational applications. He is currently a researcher at the Voice & Sound Technology Department at the Institute for Language and Speech Processing (ILSP) leading the speech synthesis team. His research interests include speech processing and applications, computational intelligence, software agents, hybrid systems and robotics.