Available online at www.sciencedirect.com

Speech Communication 49 (2007) 317–330

SPEECH COMMUNICATION

www.elsevier.com/locate/specom

Multisyn: Open-domain unit selection for the Festival speech synthesis system

Robert A.J. Clark *, Korin Richmond, Simon King

CSTR, The University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9LW, UK

Received 12 June 2006; received in revised form 25 January 2007; accepted 25 January 2007

Abstract

We present the implementation and evaluation of an open-domain unit selection speech synthesis engine designed to be flexible enough to encourage further unit selection research and allow rapid voice development by users with minimal speech synthesis knowledge and experience. We address the issues of automatically processing speech data into a usable voice using automatic segmentation techniques and how the knowledge obtained at labelling time can be exploited at synthesis time. We describe target cost and join cost implementation for such a system and describe the outcome of building voices with a number of different sized datasets. We show that, in a competitive evaluation, voices built using this technology compare favourably to other systems.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Speech synthesis; Unit selection

1. Introduction

Over the last decade, the Festival speech synthesis system (Taylor et al., 1998) has become the de facto standard free toolkit for speech synthesis research. It has also formed the starting point for at least three leading commercial systems.[1]

Until recently, Festival offered two distinct methods for concatenative synthesis: a conventional single-instance diphone-based method using an inventory containing one recording of each diphone type, and the "clunits" method (Black and Taylor, 1997) which uses an inventory of units recorded in natural sentences and performs a restricted form of unit selection.

In this paper, we introduce a third method: a general-purpose unit selection algorithm, along with the tools for building voices. The method is general-purpose because it is capable of realising open-domain voices ("clunits" performs best in limited domains, where the recordings in the inventory are from the same domain – e.g. use the same limited vocabulary and constrained syntax – as the utterances to be synthesised). We call this method "Multisyn" and it can be downloaded as part of Festival 1.95 and above from https://www.cstr.ed.ac.uk.

Unit selection speech synthesis (Black and Campbell, 1995; Hunt and Black, 1996) was proposed as a way to solve some of the problems of unnaturalness introduced by the signal processing techniques needed to produce convincing synthetic speech from a database consisting of a single example of each diphones that occurs in a language. Instead of having one example of each diphone, a number of examples in different contexts are included, and the synthesis process is formulated as a search problem. A search is performed to find the best sequence of diphones (or potentially other sized units). The goal of unit selection speech synthesis is to select a sequence of diphones which requires much less signal processing than standard diphone synthesis, or ideally no signal processing at all.

* Corresponding author. Tel.: +44 131 6511767.
E-mail addresses: robert@cstr.ed.ac.uk, Rob.Clark@ed.ac.uk (R.A.J. Clark), korin@cstr.ed.ac.uk (K. Richmond), Simon.King@ed.ac.uk (S. King).

[1] From Rhetorical Systems (now Nuance), AT&T and Cepstral.

0167-6393/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.specom.2007.01.014

There are a number of important issues to be addressed in a robust and efficient implementation of unit selection, and recent advances have lead to an improved understanding of the process. The first of these involves designing the recording script. Much of this work discusses the use of greedy algorithms to optimally select a script from a very large text corpus, examples include the work by van Santen and Buchsbaum (1997), Bozkurt et al. (2003) and Kominek and Black (2004), whilst other work discusses the theoretical and practical problems of recording the ideal dataset (Möbius, 2001).

Once a dataset has been recorded, it needs to be searched efficiently. The general search method (Hunt and Black, 1996) has been refined (e.g. Conkie, 1999; Taylor, 2000; Bulyko and Ostendorf, 2001) and complemented by other procedures for specific tasks such as limited domain speech synthesis (Black and Lenzo, 2000).

The primary goal of our Multisyn engine is to provide state-of-the-art unit selection speech synthesis within a framework that makes it easy to (semi-automatically) develop new voices, with only limited speech synthesis knowledge.

1.1. Unit selection speech synthesis

A full tutorial on unit selection speech synthesis is beyond the scope of this paper; we refer the reader to (Hunt and Black (1996)). However, we will define the terminology to be used in the rest of this paper.

Unit selection speech synthesis uses a recorded *database* (sometimes called the *inventory*) of speech. This usually consists of recordings of isolated, naturally occurring sentences (e.g. from newspaper text). The inventory along with its associated linguistic annotation is called the *voice*. *Units* are extracted from this database and concatenated to synthesise novel utterances. The unit type may be the same throughout the database (e.g. diphones), or variable (e.g. a mixture of phones, diphones, syllables, etc.). The database should contain multiple examples of each unit type.

To synthesise a novel utterance, a *target* utterance is constructed, which consists of the desired linguistic specification of the utterance: the words, the phone sequence, the syllable boundaries, placement of accents, optionally a pitch contour and segment durations, and so on. The target is constructed from the input text by the language processing front end, which is usually using some combination of rules and statistical models.

A sequence of units taken from different places in the database is then found which best matches this target. This task is performed by the unit selection *engine*. "Best matching" is measured by two costs, summed over the unit sequence. The chosen unit for a given position in the target utterance is selected from a set of available *candidate units* which may be all matching diphones (regardless of context) in the inventory, or may be a subset of those (after some pre-selection has been applied – Section 3.6).

The *join cost* estimates how well two consecutive units will join together in the large number of cases where they were not contiguous in the database and is commonly computed using only acoustic features. The *target cost* measures how well a unit matches part of the target specification, for example in terms of the constituent phones, within-syllable or within-phrase position, and is commonly computed using linguistic features. Since the join and target costs are locally computed, a Viterbi search can be used to efficiently search for the unit sequence that minimises the total cost. The details of how the join and target costs are computed vary from system to system.

1.2. Structure of this paper

Since Festival is primarily a research toolkit, this paper concentrates on explaining how Multisyn satisfies two design goals. The first goal is to provide a stable general-purpose unit selection implementation that is suitable for carrying out further research into unit selection and related techniques. The second goal is to provide the end user with a simple, mostly automatic mechanism to build their own voice for the system, requiring only limited specialist knowledge. As we shall see, this second goal means that there are times when we have employed a simple but robust technique instead of a *potentially* better, but more complex, technique. Particular attention is given to the design decisions and procedures required to build new voices.

In Section 2 we describe the design and implementation of the Multisyn unit-selection engine. The front end processes used with this engine are simply a subset of those used in the standard diphone system so are not described in this paper. We also compare and contrast the Multisyn approach to other approaches. Sections 3 and 4 discuss the requirements for the database and the process of building a voice from it respectively. In Section 5, we address the issue of automatic segmentation to phonetically label recorded speech databases. In Section 6 we discuss speech synthesis evaluation techniques and recent evaluation in which the Multisyn engine has been involved.

2. Multisyn design and implementation

The Multisyn unit selection algorithm implemented in Festival is conventional and reasonably straightforward, and follows the description in Section 1.1.

2.1. Festival's architecture

Festival is modular and uses a simple framework, commonly known as a "blackboard architecture". The system is centred on a common data structure, called the Utterance, which is passed from module to module within the system. Modules either modify existing parts – called Relations – of this Utterance structure, or add new Relations. This architecture allows users to control easily both the sequence of processes in the pipeline of modules (perhaps adding new processes) and the processing modules themselves. Multisyn is implemented as a module for Festival, and replaces a number of the modules from the pipeline for the standard diphone method, as Fig. 1 shows.

2.2. Choice of unit type

The choice of sub-word unit is influenced by a number of factors. First and foremost, the unit boundaries must be suitable concatenation points. A secondary important consideration is that it should be possible to (semi-)automatically segment the speech using standard automatic speech recognition (ASR) forced alignment techniques.

Of all the possible unit types, including phones, half phones, diphones, syllables or larger units [e.g., units matching prosodic structures in (Taylor, 2000)], we opted to use diphones. These satisfy both requirements above: diphone boundaries can be easily derived from phone alignments. Implementation of the Viterbi search using fixed-size units is also considerably simpler than for variable-sized units (see Section 3.6). Although using smaller units such as half phones implicitly helps alleviate some problems of data sparsity, it also makes the Viterbi search for units far more computationally expensive. We have opted for the advantage of the more efficient search possible with the diphone base type, and have instead chosen to implement certain strategies for dealing with special cases of missing units; for example, backing-off to a different unit type, or the possibility of extending the margins of the units

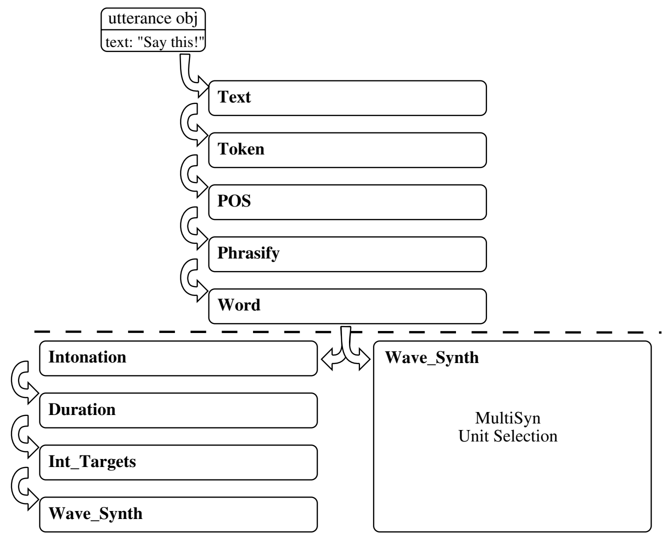

Fig. 1. The relationship of Multisyn (right-hand branch) to standard single-instance diphone synthesis (left-hand branch) within Festival's blackboard system architecture. MultiSyn is implemented as a module within the Festival speech synthesis system pipeline, as an alternative to the standard prosody prediction and diphone concatenative synthesis modules, while still making use of the other front-end linguistic processing modules. The Text, Token, POS, Phrasify and Word modules normalise the text and produce a linguistically annotated segment sequence, the resulting output is then passed to the waveform synthesis modules.

either side of the missing unit and making a join at a phone boundary. These methods are described in more detail in Section 3.7.

We decided not to use explicitly variable sized units and believe that the selection of such units should result from the search (i.e. through selection of contiguous sequences of diphones from the inventory), rather than be predefined.

Multisyn is implemented in such a way that using units other than diphones would only require a small amount of programming effort. We are currently considering implementing other unit types as a partial solution to the problem of cross-language diphone coverage (Black and Lenzo, 2004).

2.3. Comparison with other methods

A comparison between the Multisyn engine and two of the first unit selection implementations – CHATR (Hunt and Black, 1996) and Festival's "clunits" method (Black and Taylor, 1997) – is useful to clarify how Multisyn differs from other techniques.

The two major differences between clunits and Multisyn are the unit type that is used and the nature of the target cost employed to determine how good a given candidate unit is.

Multisyn by default exclusively uses diphone sized units, whereas clunits exclusively uses phone sized units. There are advantages and disadvantages of each approach. The major advantage of using phone units is that each unit can be fully described in terms of the features representing a single phone, whereas diphones require twice as many features to describe them, and questions like "is this diphone stressed?" has answers 'yes', 'no' and 'partially' rather than just 'yes' and 'no' which answer the same question regarding a phone. Additionally, the resulting number of units in the inventory squares. The clear disadvantage of using phone units is the loss of context and although clunits employs a number of techniques to take context into consideration, clunits combines acoustic information from the neighbouring segments in the acoustic representation of the unit, and additionally uses optimal coupling to find the best possible join between any two units (see Black and Taylor (1997) for more details) bad joins arise, particularly as the size of the unit inventory increases and the variation in the context of available units increases.

The second major difference between clunits and Multisyn is the target cost implementation. Multisyn implements a direct feature-based target cost, where clunits uses feature information to predict acoustic parameters which are used as the basis for the target cost. The Multisyn approach scores units based upon matches in their linguistic context, whereas the clunits approach uses the linguistic features to predict a unit's gross acoustic properties and then performs the scoring in acoustic space. This is the main problem of the clunits approach: it uses just a single vector of values to describe the complex acoustic properties of a unit.

CHATR (Hunt and Black, 1996) differs from Multisyn in two quite specific ways which we believe make Multisyn both more powerful and more efficient. Firstly, again, CHATR's default unit type is the phone which brings with it some of same problems that clunits suffers. The second difference is that the Multisyn specification of a target unit does not attempt to include information about the predicted acoustic properties of the target. Instead, the prediction of acoustic properties from linguistic features is implicit in the unit selection procedure itself.

3. Database

3.1. Coverage

Obtaining the necessary coverage for a diphone inventory is hard because context must be taken into account (van Santen, 1997; van Santen and Buchsbaum, 1997; Beutnagel and Conkie, 1999; Black and Lenzo, 2001; Möbius, 2001). Coverage of as many *diphone-in-context* types as possible is desirable but very difficult to achieve.

The number of possible contexts will be very large, because the context is usually specified in terms of several linguistic features, such as position-in-syllable, -word and -phrase, stress, part-of-speech, and left and right phonetic context. Each of these can take multiple values (e.g. lexical stress can take the values unstressed, primary, secondary and perhaps even tertiary stress), which means the number of context dependent unit types rapidly becomes very large when even just a few features are used to define unit context.

It is impossible, in practice, to include examples of all desired diphones-in-context in the inventory and a compromise must be made. In Section 4.1 we discuss how the sentences to be recorded for the inventory are selected in order to make a reasonable compromise. In Section 3.7, we describe two techniques to address the problem of missing units (which occurs in even the most carefully designed inventory).

3.2. Automatic segmentation

The context-sensitive nature of diphones, and the consequentially large number of unit types requiring coverage, mean that a large inventory of speech is required. However, this is outweighed by the possibility of completely automatic segmentation using standard Hidden Markov Models-of-phones ASR techniques. It is easier to collect a large data set than it is to hand-label a small dataset; indeed, automatic labelling accuracy should improve as more data is available to train the HMMs. The automatic segmentation process is discussed in detail in Section 5.

3.3. Target utterance specification

A target utterance structure is constructed from text (or marked-up text) input. In single-instance diphone-based systems, extensive use is made of rules and models such as CARTs to predict information about segment duration, f0 and so on. Substantial linguistic resources (e.g. hand-labelled speech data) are required to train these models.

One of the advantages of unit selection synthesis is that much of this information is not necessarily required. If available, it can certainly be incorporated into the unit selection procedure, but if it is not available then unit selection can proceed without it. This is because unit selection is guided by the target cost and the join cost, and these costs can be formulated in any way one desires, using whatever information is available in the input or is marked up in the voice (the annotated inventory).

In Multisyn, the default join cost uses solely acoustic properties of the candidate units (e.g. spectral information at concatenation points). Meanwhile, the default target cost, discussed in more detail in Section 3.8, combines the scores of a number of comparisons of predominantly linguistic features (e.g. the labels on the annotated inventory and the features derived from the input text). The cost components based on linguistic features are comparator functions which compare linguistic attributes of the target unit with each candidate. There are, however, additional non-comparator target cost components which make use of information derived from the acoustic signal: duration and f0. These additional components are special cases which complement the way in which the other target cost components are used. They are designed to lessen the impact of faulty automatic labelling in the voice inventory. The duration component penalises candidate units on the basis of comparison with the distribution of durations for a given unit type at voice building time. The f0 component heuristically penalises candidate units which appear to be either wrongly voiced or unvoiced based on their identity and phonetic context.

Note that this use of f0 and duration in the default Multisyn target cost differs significantly from their more usual use in unit selection target costs generally. In many systems, a prosody module in the speech synthesiser will estimate duration and f0 values on the basis of the linguistic front-end processing, and these are then used as explicit target values in the unit selection process. However, in the default Multisyn implementation, we have taken the approach that many properties of the target speech, including segment durations and prosody, do not need to be explicitly predicted; instead, the natural properties of the units in the database are used. In effect, the target cost (Section 3.8), by requiring that certain contextual features of the selected units match, has replaced the explicit predictive models of duration and intonation with an implicit model based on the natural characteristics of the context features used. Given a good target cost and a large inventory, this method has the potential to outperform explicit models, and has the added advantages that the selected units will have appropriate values for other acoustic properties (e.g. amplitude and spectral quality) and will require little or no modification by signal processing. This is the strategy used in Multisyn;

duration and intonation models are not currently used because none of our available models perform as well as using the natural prosody of the selected units.

To ensure that the prosody of selected units is appropriate, the contextual factors used in the target cost must include those that influence duration and intonation. In English at least, lexical stress is probably the most important factor, although other prosodic distinctions like phrase position are important too.

3.4. Expressive speech

There are situations where a default, neutral intonation is not appropriate: for example, in dialogue contexts where a contrastive tune is needed with specific words accented or de-accented to convey a particular meaning. Here, prosodic mark-up present in the input text can be used by the target cost to influence the choice of units, preferring those which carry appropriate prosody. However, this requires corresponding information to be present in the database. Automatic prosodic annotation of the database is difficult and not necessarily reliable, although it can work reasonably well in limited domains where a portion of the database which is *in domain* can be hand annotated quite easily. Alternatively, it may be possible to automatically generate annotation if the text is the output of a natural language generation system.

3.5. Modest requirements for linguistic resources

The only models that are needed are a simple phrasing model (e.g. trivial rules which use the punctuation in the input text) and a pronunciation lexicon or other grapheme-to-phoneme conversion method. Neither of these necessarily require large amounts of hand-labelled data during development.

This is particularly advantageous when developing voices in a new language, because the only new components required are a pronunciation lexicon and/or letter to sound rules. A simple part of speech tagger that can make a content/function word distinction is also useful as this can help contribute to selecting units with the correct stress, but this is not essential as part of the first attempt at a new language. This is not to say that developing a voice in a new language is necessarily easy: creating the pronunciation lexicon for a language with few existing resources is not trivial.

In order to construct the voice, a large set of sentences is required, from which a subset is selected that provides the best possible diphone-in-context coverage. We discuss this in Section 4.1.

3.6. Candidate pre-selection and beam pruning during search

The output of the front end is a target phone sequence with an appropriate linguistic structure attached. In Festival, this structure is stored as a number of parallel streams, represented as heterogeneous relation graphs (Taylor et al., 1998). The structure includes annotation of syllabic structure, phrasing and part of speech tagging.

The target phone sequence is first converted to a sequence of diphone units; a list of candidates for each target unit is retrieved from the inventory and the unit selection engine then proceeds to search for the candidate unit sequence with the lowest cost.

An optional pre-selection step can be used immediately after retrieving the candidates from the inventory, to restrict the number of candidates per target diphone. Methods of pre-selection vary from the complex, such as phonological structure matching (Taylor, 2000), to the simple, such as only including units which are appropriately stressed/unstressed. Pre-selection can speed up the search substantially by restricting the search space and by reducing the number of join cost calculations that need to be performed (which can otherwise be very large).

However, for a research system such as Festival, the use of pre-selection is less attractive than for a commercial system. Pre-selection is an attempt to simulate (in a computationally cheap way) the complex interaction between target and join costs during the search procedure. Using pre-selection further complicates these interactions. In the same way that debugging the implementation of an ASR system is made much harder when there are pruning errors, candidate pre-selection can make analysis of target or join cost behaviour very difficult.

If speed of synthesis is important, this can be achieved using beam pruning during the search. For example, if stress is deemed to be sufficiently important that units carrying the wrong level of stress should not be used, then a suitably large weight on the stress component of the target cost in conjunction with beam pruning can ensure that if one suitably stressed candidate is available then all unsuitably stressed candidates are not considered. The advantage of this is clearly that if there are no stressed candidates available then unstressed ones will automatically be considered.

3.7. Missing units

We have so far assumed that the candidate list for each target diphone contains at least one suitable candidate. For this to be true, at least one token of each diphone type needs to be present in the inventory. Unfortunately, however, this may not be the case.

Even though great care may be taken at the stage of designing a voice database, it is nevertheless rather difficult to ensure that there are absolutely no missing diphones. Problems arise, for example, when the lexicon used to design the database (Section 4.2.2) is not precisely the same as the one used within the synthesiser, or when the lexicon or post-lexical rules have been modified after the data was recorded, to match idiosyncrasies of the speaker, or when the speaker has not uttered certain recording script prompts entirely as predicted during the text selection process. In other cases, the voice builder may have little or no control over the speech data used to build the voice, for example where a voice is built from spontaneous speech or data recorded by someone else.

Therefore, it is important to have a strategy for dealing with the problem of unit types required at synthesis time which are not present in the voice database inventory. We have implemented two alternatives for dealing with this situation: back-off and phone boundary joins.

3.7.1. Backing-off

One way of dealing with cases where an exact match of unit is not found in the unit inventory is to identify alternative suitable candidates. Multisyn does this by *backing-off* the target specification. If a diphone cannot be found, an ordered list of possible substitution rules is consulted in an attempt to find an appropriate alternative diphone. This list is generally associated with a particular lexicon, and so becomes language or dialect specific.

In initial experiments, we used a back-off procedure which altered the target phone sequence to find not just a replacement for the missing diphone, but then to substitute adjacent diphones to preserve consistency of the overall phone sequence. For example, to synthesise part of the word "team" the diphone sequence [t]-[i] [i]-[m] is required. If the diphone [t]-[i] is missing and consequently substituted by the diphone [t]-[ə], an attempt would then be made to substitute the target diphone [i]-[m] with [ə]-[m] to keep the phone sequence consistent.

It quickly became apparent that this was a difficult search problem and that, unless the substitution rules were written very carefully, it was difficult to obtain a suitable substitute phone sequence. The procedure was therefore simplified and the current back-off procedure does not correct adjoining diphones. Any substitution that occurs means that there will be an inconsistency in the diphone sequence. The join cost is then relied upon to find smoothly joining units.

Obvious substitution rules to use include: use reduced vowels instead of full vowels (in which case there are probably instances of the full vowels and reduced vowels which are spectrally close enough to join reasonably well, since vowel reduction is a continuum); substitution of [n] to replace [^{ə}n], where there will be little difference at the join point.

3.7.2. Phone joins

A second method implemented in Multisyn for dealing with cases of missing diphones works by extending the margins of the units either side of the missing unit to make a join at a phone boundary. This method obviously is only suitable in cases where there are candidates present in the inventory for the immediate right- and left-neighbouring diphones in the target specification. Fig. 2 illustrates how this method works in the case of a missing [æ]-[t] diphone when synthesising the word "cats".

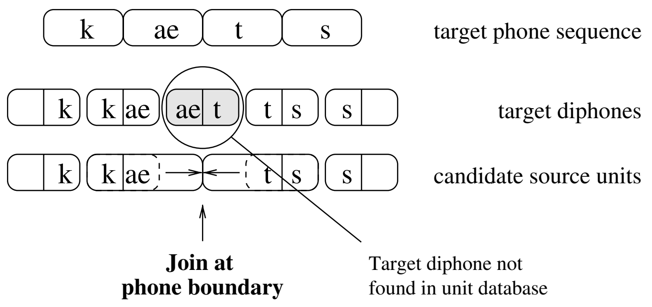

Fig. 2. An illustration of using phone boundary joins to deal with isolated instances in the target diphone sequence which are missing in the voice database. In this example, the [æ]-[t] diphone required to synthesise the word "cats" is missing. In order to compensate for this, the margins of the directly neighbouring diphone source units can be extended as shown. This results in the inclusion of whole [æ] and [t] phones from the voice database, with the join at the phone boundary between them instead of the usual diphone boundary.

An initial check of the target sequence is made for diphone units not available in the voice database. For isolated missing units, the concatenation point of all candidate source units is taken to be at the following phone boundary for left neighbours and at the preceding phone boundary for right neighbours. The Viterbi search is then run in the usual way, such that join costs are calculated at all potential concatenation points, including any at phone boundaries. The target cost is not calculated for the missing target diphone unit. This technique is only employed for isolated missing diphones.

While phone boundaries do not always have the same attractive properties in terms of concatenation as the centres of phones used in diphone joins, this method does have certain advantages. For example, it is suitable for a voice which can switch between languages mid-sentence. Diphones consisting of phonemes from different languages will probably not be available in the database; phone boundary joins are a simple and effective solution (Kurtić, 2004).

3.8. Target cost

To facilitate ongoing research on target costs (e.g. Hofer et al., 2005) and join costs (e.g. Vepa and King, 2004), the implementation of these costs in Multisyn is designed to allow flexibility. Hence, new cost functions can easily be added by the user.

The target cost is configured at the voice level, allowing different voices to use different target costs. Individual target cost implementations are derived from a base C++ class which provides an API, a core set of comparator functions (which compare features of the target to those of a candidate) and a mechanism for weighting these functions (the target cost is thus a weighted sum of comparator functions). Additional comparator functions can easily be added to a target cost derived by the user from the base cost class or the derived target cost can completely bypass the mechanisms in the base class. The system provides a number of default target costs which can be used at run-time by any voice. These also provide reference implementations for users wishing to implement their own target cost. A derived class is also supplied which allows the target cost to be implemented in Scheme rather than C++. This facilitates rapid prototyping of novel target cost functions, without the need to recompile.

Designing a target cost involves deciding which characteristics of the target should be considered (the choice is of course restricted to those characteristics which are known for both target and candidates), how each characteristic is to be compared (i.e. how the difference between target and candidate should be converted into a scalar) and how these costs should be weighted relative to one another. The default target cost has been constructed by hand, based on developer intuition. Stress and phrase position are generally regarded as the most important characteristics; components are also included which penalise candidates that have been labelled as durational outliers or as badly pitch-marked.

Other systems (e.g. Hunt and Black, 1996; Syrdal et al., 2000) have described methods for automatically training the weights on the individual components of the target cost. These methods are not implemented in Multisyn because it appears that the gains that can be achieved over heuristic methods are currently only slight; even with no target cost specified, an intelligible voice can be produced from a well-designed and accurately labelled database. Preliminary results from our continuing research into human perception of speech synthesis experiments suggest that keeping the number of audible joins to a minimum and ensuring a low overall target cost is far more important than having a target cost with carefully weighted sub-components which can subtly discern between different levels of 'badness' between units. The current default target cost is structured as shown in Table 1. Target cost sub-components are generally discrete and take values of 0 or 1. In some cases, values of 0, 0.5 or 1 are used: 0.5 is applied for each of the two half phones constituting the target diphone.

Table 1
Component cost functions for diphones, used in the current default target specification

Feature	Weight	Description
Phrase	15	Position in phrase is correct (categories: initial medial or final)
Stress	10	Stress is correct
POS	6	Part of speech is correct (categories: POS uses only five tags, nouns, verbs, modifiers, function words and other)
Syllable	5	Position in syllable is correct (categories: initial, medial, final and between words)
Word	5	Position in word is correct (categories: initial, medial, final, between words)
Left	4	Left phonetic context matches
Right	3	Right phonetic context matches

Overall target cost is the weighted sum of the above components normalised to be in the range [0, 1].

The use of only linguistic based features taking discreet values in this way generally works well, although a few problems remain. We have been unable to develop a sub-component to deal with prosody and accent, that performs well in situations where prosody other than neutral declarative sentences are required. This is primarily because of the difficulty of constructing an accurate accent predictor for the recorded database and for use during unit selection. Another issue is that it appears to be more important to get certain units "right" than others. For example, if the last diphone in a phrase is not taken from phrase-final position, the resulting speech usually sounds very unnatural. Simply imposing a high penalty cost on such units does not work well: if the offending unit is at the end of a long contiguous sequence of units selected from the database, the zero total join cost of this sequence can offset the penalty cost.

3.9. Join cost

Like the target cost, the join cost is implemented as a C++ class. The default join cost employs three equally weighted sub-components for f0, log energy and spectral mismatches. Spectral discontinuity is estimated by calculating the Euclidean distance between two vectors of 12 MFCCs from either side of a potential join point, as the MFCCs are usually mean/variance normalised first, this is effectively a Mahalanobis distance with diagonal covariance. For energy, the magnitude of the difference across the join point is used. For f0, joins between voiced segments use the magnitude of the difference across the join point; joins between unvoiced segments incur zero cost; joins between a voiced segment and an unvoiced segment incur a large penalty, equivalent to a mismatch between voiced segments of four standard deviations of the speaker's pitch range. Delta and delta–delta derived features are not currently used in the default join cost. These three components are normalised (using means and variances calculated across the entire voice database) to lie within the range [0, 1] during voice building.

The weightings for the current target and join costs have been set heuristically to provide a baseline acceptable performance, but these can easily be changed to values based on statistical training or perceptual evaluation, should data be available.

4. Voice building

At the core of any good unit selection speech synthesiser is the *voice*: a database of speech from a single speaker, annotated with time-aligned phonetic labels and additional linguistic information. In designing a voice, there are two intertwined questions that must be addressed: How big should it be? What sort of data should it contain? While there are no simple answers to these questions, there are constraints which, in practice, will guide the design of a particular dataset. We address this issue here by discussing existing data sets that we have recorded and comparing them with those recorded by others. To provide general guidance for database design, we will cover some of the criteria that we feel are most important to keep in mind when designing a new voice.

4.1. Voice database design

There are two conflicting factors which govern voice database design: extensive diphone coverage demands a large database, yet there are practical difficulties in obtaining consistent recordings from the speaker over multiple sessions. We consider the common requirements for the material in the database to include the following:

(1) wide and well-balanced phonetic coverage of context-dependent diphones;
(2) phrase-final (and other intonational) coverage;
(3) common structures and idioms, such as lists, dates, etc.;
(4) names and other material common to the target domain.

As noted, the most important issue is general coverage of context-dependent diphones, where the context is loosely related to the different criteria that make up the target cost. The ARCTIC (Kominek and Black, 2004) voice size of around 36,000 phones seems suitable for a basic voice. However, we believe that for a good voice these 36,000 phones should form the base to which additional material need to be added to fulfil specific requirements.

The second requirement, although part of the context described above, is sufficiently important that it is mentioned separately. Using non-phrase final diphones in phrase final position can make an otherwise good utterance sound completely unnatural. With this in mind it is good practice to ensure that there is a plentiful supply of individual diphones in phrase final (and probably also "close to phrase final") position. For English voices, special attention should be paid to ensuring there is sufficient coverage of question intonation, where a pitch rise occurs at the end of the utterance. Other languages may have other intonation patterns that need to be included.

List structures, times and dates, Zip/Post codes, telephone numbers etc. all have special structures and synthesised examples of them tend to sound unnatural when there are not sufficient examples of these types of structures in the database from which to select units. It is prudent to incorporate a large block of such data if the resulting voice is required to synthesise this kind of material.

Finally, since the construction of the script is a greedy process, the order in which material is added must be considered. It is wise to address the first of the issues (phonetic coverage) *last*. The database design process should begin with the inclusion of domain-specific material, dates, currencies, names and so on. Once the phonetic content of this initial, mandatory material is measured, it can form the starting point of an automatic process for selecting supplementary material to ensure full phonetic coverage.

4.1.1. Requirements for the database and text selection

Units are required in many different *contexts*. The distribution of diphone occurrence with respect to context means that a large number of diphone-in-context types occur very infrequently, making it difficult to design a compact inventory, while any given sentence to be synthesised has a reasonably high probability of containing at least one rare diphone (Möbius, 2001). This highlights the need for a mechanism, at synthesis time, to choose a suitable diphone where the ideal diphone is unavailable. The linguistic features used to represent context during script design should match those used by the target cost and those that have high weights in the target cost should be considered more important by the text selection process.

4.1.2. Post-recording considerations

Diphones supposedly covered by the script can be found to be missing after the recording has taken place for a number of reasons. The speaker may have spoken a word with a different pronunciation (assuming the labelling for this word has been adjusted to match what the speaker actually said), or may have left a pause between words in an unpredicted place. In situations where an existing dataset has been used to build a voice, the planned coverage cannot be controlled at all (examples of using existing data are described in Section 4.2).

4.2. Analysis of existing voices

Table 2 lists four Multisyn voices built from various data sets. Run times are relative to the fsew0 voice and indicate the complexity of the search required for each voice. All of these voices were built using the Multisyn voice building tools, distributed with Festival 2.

4.2.1. "fsew0": small database, not synthesis-specific, open domain

The "fsew0" voice was built from the MOCHA data set of the same name (Wrench, 2001). It comprises readings of 450 TIMIT sentences (Garofolo, 1988) plus an additional 10 sentences covering British English phonetic combinations and was not designed specifically for speech synthesis (although the TIMIT sentences were designed for

Table 2
Comparison of voice size and computational load

Voice	Phones	Relative run time
nina	175 000	9
jon	60 000	3
awb (ARCTIC)	36 000	2
fsew0 (MOCHA)	14 000	1

maximum phonetic richness). The data were recorded in a Carstens AG100 EMA machine.

The voice built from this data contains about 14,000 phones (approximately 30 min of speech including some silence) and was built primarily to evaluate the usefulness of articulatory information as a contribution to the join cost. The obstructions of the instrumentation in the mouth make the segmental quality of the original speech and of speech synthesised with this voice somewhat unnatural. Along with its small size, this contributes to the voice's low perceived quality. We consider that the amount of data in this voice is simply insufficient to achieve an acceptable level of naturalness. The high proportion of missing diphones drastically lowers the voice's intelligibility.

4.2.2. *"awb": medium-sized database, synthesis-specific, open domain*

The "awb" voice was built from the ARCTIC (Kominek and Black, 2004) database of the same name, which was specifically designed for speech synthesis using text selected from out-of-copyright books. This voice contains around 36,000 phones (1.4 h of speech). Its performance is somewhat varied, being mostly of reasonable quality, but often suffering from intelligibility problems. Some of these problems may be related to our segmentation of the data (using the forced alignment procedure of the Multisyn voice building tools) and others may be due to missing diphones in some contexts. This latter point results from the fact that the criteria used by the database designers to ensure diphone coverage may not match the criteria that we assume at synthesis time (i.e. those used in our target cost). We believe that the database used in this voice is of the minimum possible size for reasonable performance.

Using a Scottish pronunciation lexicon (the speaker "awb" is Scottish) rather than an American one (for which the diphone coverage was designed) yielded slight improvements, both in automatic segmentation and resulting synthesis, but the differences are not conclusive due to the coverage issues arising from the small database size. This text that this database is built from is the same as used for the *Blizzard* Challenge (Black and Tokuda, 2005) discussed in Section 6.1.

4.2.3. *"jon": large database, synthesis-specific, limited domain*

The "jon" voice was designed by us as a limited domain voice for the COMIC project (Foster et al., 2005). The dataset contains two parts: a base set of around 650 sentences designed to provide a basic level of diphone coverage and 300 domain-specific sentences that deal with the subject of bathroom design. The base set speech is of a similar size to the awb set and the domain-specific set is similar in size to the fsew0 set.

The performance of this voice is generally excellent when generating in-domain sentences, but (as expected) quality is dramatically lower (worse than "awb") on out-of-domain sentences. The primary reason for this is thought to be speech rate problems. The speaker used for the "jon" voice spoke at a faster rate than the other voices described here. This means that there are many very short, elided or deleted segments. This caused problems with the automatic labelling. A large number of very short segment labels result from a combination of actual short segments and incorrectly placed labels (e.g. labels for segments that are actually missing). If two consecutive segments are sufficiently short then no pitch marks are present in the corresponding diphone. It is then not possible to use this diphone because the pitch-synchronous waveform resynthesis uses windows centred on pitch marks and extend (asymmetrically) in time from the preceding pitch mark to the following one (i.e. they have a duration of 2 pitch periods).

4.2.4. *"nina": large database, synthesis-specific, open domain*

The "nina" voice is approximately three times the size of "jon" and five times the size of "awb". This results in a quite noticeable improvement in the quality of the output. This voice is, however, far from perfect. The text selection procedure used by us to design the recorded prompts only used sentences from newspaper texts. While this achieves a good level of general diphone coverage (and the voice performs well on text that is similar to this domain) the voice often performs badly when specific grammatical structures, not commonly found in newspaper text, are encountered, such as lists, times and dates. In these cases, even when the segmental quality is good, inappropriate prosody reduces the perceived overall quality. More importantly, this voice synthesises question intonation very badly because there were very few questions (with rising utterance-final pitch) in the recorded speech.

4.3. *Analysis of the approach in general*

The approach seems ideal for the rapid development of new voices, and also a useful teaching tool, the system is currently used by one of the authors for an postgraduate course in speech synthesis where students with little previous experience record themselves and create their own voice. While the approach in some circumstances produces results that may not sound as natural as other more complex systems, the overall simplicity of Multisyn and its minimal requirements for making a good quality intelligible voice more than make up for this. Our experience also suggests that the difference between a good voice and an excellent one is more often than not related to the number of hours spent manually cleaning up the data where the automatic techniques (segmentation, pitch marking, etc.) have not produced perfect results.

5. Automatic segmentation techniques

The main issue in obtaining a phonetic labelling for the recorded speech data is not necessarily pin-point precision of the phone boundary times but a combination of reasonably accurate boundary times, the correct choice of the segment labels themselves and a knowledge of where labels that are suspected to be inaccurate are in the database. Assuming that the synthesis unit is the diphone, the labels do not need to be placed precisely at phone boundaries, since these labels are only used to derive the diphone boundaries, which are placed midway between phone boundaries for most segment types (except diphthongs, whose cut points are at 25%, and stops and affricates, whose cut points are at the end of the closure portion – see Section 5.6). Joins are made at diphone boundaries because the spectrum is expected to be locally static compared with phone boundaries. Small inaccuracies in the phone boundary positions will still result in diphone boundaries that fall within these static regions most of the time. With this in mind, it seems that there is little value in developing methods for high precision placement of phone boundaries. Techniques such as optimal coupling (Conkie and Isard, 1996) can adjust the actual cut points at synthesis time (typically by minimising the local join cost, after the unit sequence has been selected), compensating for some inaccuracy in label placement.

Determining an appropriate phonetic label sequence for each utterance in the database is a major challenge and there are two aspects to this problem. First, the general problem already faced in text-to-speech synthesis of converting a word string into a phoneme sequence, accounting for effects of connected speech, the speaker's accent and so on. The second problem is that the recorded speaker may not have produced speech which precisely matches this predicted phoneme sequence.

5.1. Converting text to phonemes

To address the issue of pronunciation variation arising from a speaker's specific accent, we use an accent-independent keyword lexicon (Wells, 1982). From the underlying *Unisyn* lexicon (Fitt and Isard, 1999) a surface-form accent-specific lexicon is generated for a particular speaker. This can be tailored to the individual speaker (using rules) if necessary.

The resulting surface-form lexicon will contain multiple pronunciations for some words, for example stressed and reduced forms of function words. Festival is currently forced to choose one pronunciation at synthesis time, rather than deciding which pronunciation to use on the basis of the units available.

Once an accent-specific lexicon is generated for a particular voice, an initial label sequence for the forced alignment is produced using the linguistic analysis phase of the text-to-speech synthesis process – i.e. lexical lookup, letter-to-sound, and post-lexical rules. A few modifications are then made to this sequence to facilitate more accurate forced alignment. Stop and affricate labels are split into two parts (closure and release). This will later allow diphone cut points to be placed at the end of the closure portion. Sentence-initial and -final silences and optional *short pauses* between words are added. Such optional inter-word short pauses are common practice in ASR and will be skipped during alignment if no silence is present.

5.2. Pronunciation variation in the recorded speech

Variations in the pronunciation of words in the recorded database fall into two categories: expected pronunciation variation and unexpected pronunciation variation. *Expected* variation is described in the lexicon, such as vowel reduction or deletion, or alternate pronunciations (e.g. 'either' [iðə/ [aɪðə] and 'against' [əgɛnst]/[əgeɪnst]. *Unexpected* variation occurs where the speaker pronounces a word in an unexpected and often unpredictable way. They may misread the script or they may pronounce an uncommon word incorrectly, which most often occurs with names. We currently deal with *unexpected* pronunciation variation by altering the script (after recording but before phonetic labelling) to match what the speaker actually said. Where a word has been misread, we update the script, but where a word is systematically pronounced in an idiosyncratic way, we alter the lexicon used for phonetic labelling.

5.3. Handling conflict between the predicted labels and what the speaker said

Typically, a synthesiser front end predicts a single phonetic sequence for a given utterance. If this sequence is used to label a recorded utterance there is a potential mismatch between the predicted labels and what the speaker actually said. Even if the synthesiser attempts to *predict* expected pronunciation variation (e.g. which vowels will be reduced) it is unlikely to be an exact match to what the speaker said.

There are two extremes when labelling the database: matching precisely what the speaker actually said, or matching precisely what the synthesiser predicted. Neither is entirely satisfactory.

Labelling the database with phonetic sequences that match what the synthesiser would predict at synthesis time is certainly the easiest option and requires little work. This guarantees that any word sequence present in the database can be synthesised very well, and it does not matter that the phonetic labels on the database do not accurately reflect the actual speech signal. However, problems arise below the word level, when sub-word units are concatenated. Wrongly labelled units will result in synthetic speech which has the wrong phonetic content and will probably have more bad joins (because mismatching diphones are being concatenated).

Labelling the database with a phonetic sequence which matches what the speaker said, will eliminate that problem, but not only is this a much harder task, the segment sequence predicted for a word at synthesis time may no longer match the labelling of tokens of that word in the database. This will result in it being synthesised from non-sequential units rather than the sequential ones, which is likely to sound worse.

The current approach in the Multisyn voice building tools is to find a compromise between these two extremes. We now describe the basic process of forced alignment, for the case where a single phonetic label sequence – predicted by the synthesiser without reference to the speech signal – has been determined for each recorded sentence. We then describe how some decisions on the label sequence can be deferred until during the alignment procedure: this allows labels to more closely match what was actually said without drifting too far from the predicted sequence.

We are currently still investigating how best to reconcile the differences between changed database pronunciations and runtime pronunciations. The current implementation allow a parameter to be set which decides whether reduced vowels or full vowels should be specified, this affects the utterance as a whole rather than allowing differences within an utterance as the techniques proposed by Bennett and Black (2005) and Hamza et al. (2004) would.

5.4. Aligning the phonetic labels with the speech

Given a single phonetic label sequence, there are well-established methods to find an alignment with the corresponding speech signal using standard automatic speech recognition techniques.

Since there are guaranteed, by design, to be many examples of all phones in the speech data, it is straightforward to train a set of speaker-specific hidden Markov models (HMMs). The HTK toolkit is used (Young et al., 2002) and we begin by taking three-state monophone models with a left-to-right topology, observations are modelled by mixture-of-Gaussians state output PDFs (trained initially with only a single component).

The speech is parameterised as 12 Mel-scale cepstral coefficients plus energy, deltas and delta deltas (a total of 39 features). A relatively short window size of 10 ms is used with a short 2 ms shift. Initial results suggested that this generated more accurate and consistent boundary positions and fewer gross labelling errors than using standard values of around 25 ms and 10 ms, respectively.

We do not use triphone models because the performance of monophone models is deemed good enough not to warrant the significant additional complexity of building triphone models. The models are only required to produce a *consistent* alignment (i.e. the positions of the label boundaries are relatively the same across the database, even if they do not necessarily match what are traditionally considered phone boundaries) rather than perform speech recognition.

Forced alignment using these models with a single, known label sequence is trivial and computationally very fast. Note that the training and "testing" data are one and the same thing: the complete set of voice data.

5.5. Making label decisions during forced alignment

If one were to attempt a "pure" phonetic labelling of the speech, without reference to the known word string, this would be achieved using a phone recogniser constructed from the same HMMs as for the forced alignment along with a "phone loop" grammar. However, the expected accuracy of such a procedure is unlikely to be high enough for our purposes. Therefore, a precise phonetic labelling based only on the speech waveform is not possible.

To achieve the desired compromise between predicted and actual phone labels, for each sentence in the data a phone lattice is constructed which includes any plausible pronunciation variation. Currently, the only pronunciation variations allowed are vowel reductions. This procedure is similar to the procedure described by Bennett and Black (2005), although whereas they discuss only function words, we allow reduction to potentially occur for any phonologically reducible vowels in both function words or in the unstressed syllables of contend words. The design of the lattice is currently being extended to include other types of variation. Indeed, if a lexicon listing possible pronunciation variants for all words was available, these could easily be incorporated. This lattice is much more constrained than the "phone loop" grammar and therefore we can expect alignment high accuracy.

The lattice is initially aligned with the speech using a set of HMMs that have been trained on the single transcriptions. The result is an intermediate transcription (the most likely path through the lattice). The models are then retrained using this transcription. During this second phase of training, the number of components in the state output densities is gradually incremented up to eight components (a number determined empirically) using HTK's standard "mixing up" procedure (Young et al., 2002). A forced alignment using the final models then produces the labelling for the voice database.

5.6. Post-alignment processing

Once the alignment is done, the label times are reconciled with the linguistic structure generated by the synthesiser. This process deals with any inter-word short pauses detected during the alignment, substitutions (i.e. vowel reductions), and the merger of the closure and release portions of stops and affricates back into a single label. Substitutions are marked in the linguistic structure (for possible later use in unit selection) and the end of the closure portion of stops and affricates is noted for later use as the diphone join point for these segments.

Other information is associated with individual phones in the linguistic structure to enable the target cost to incorporate a component which deals with suspected bad labelling. This information includes a normalised version of the HMM log likelihood score for each segment and a flag which marks segments which are too short to have meaningful pitch-marking.

The distributions of the duration of each phone type are also analysed and any outliers are marked as such. This information is made available at synthesis time to guide the unit selection search.

The result of the alignment procedure is a segmental labelling that is consistent and sufficiently accurate (in time) for deriving diphone cut points. A formal evaluation of the accuracy of the segmental labelling is costly since manually verified reference labels have to be created. It may also be unnecessary, because consistency and the ability to know which label times may be inaccurate are more important than label times accurate to the nearest millisecond. Makashay et al. (2000) show that automatic segmentation is potentially better than manual segmentation for speech synthesis, which suggests that the inconsistency in hand-labelled data makes it an inappropriate baseline.

However, hand correction of the automatically aligned labels may still be desirable. Our experience is that the alignment is always poor for some fraction of the sentences. Much of the time this means labels with durations that are clearly outliers for that segment class; these can be easily spotted and then either removed or flagged as bad units. It is an open research question whether such bad units can still be used for synthesis (provided their left and right neighbours are also used).

A sizable proportion of gross errors that occur are caused by the speaker saying a word sequence that does not match that predicted by the synthesiser, particularly for expansions of acronyms and numerals. Fully normalising the script, including expanding all abbreviations, dates, etc. into unambiguous word sequences is the most reliable solution to this problem.

6. Evaluation

Evaluation of a speech synthesis implementation is rather difficult to perform, especially since the most obvious comparison is with various other speech synthesiser implementations rather than between other techniques within the same system. For this reason, we are fortunate to benefit from the recent inception of the *Blizzard* Speech Synthesis Challenge, an initiative for the competitive evaluation of speech synthesis systems.

6.1. The Blizzard Challenge

The first *Blizzard* Challenge (Black and Tokuda, 2005) was held in 2005, and reported in a special session of the Interspeech conference in Lisbon, Portugal. For this competition, a total of four pre-recorded speech databases were released. Each entrant was asked to build four voices for their speech synthesis system. An unseen test set of sentences was then released, and entrants were required to use the voices they had built to synthesise the test sentences and send the waveforms to the organisers for perceptual testing. Two of the four datasets were released at short notice before the release of the test set, thus minimising the possibility for hand tuning and forcing entrants to rely upon automated voice building techniques.

Listening tests were carried out with three groups of subjects: web-based listening tests were undertaken both by 50 researchers in the speech technology field and by 60 non-speech technology-related people, while a set of 58 undergraduate students were employed in listening tests under more rigorous laboratory conditions (Bennett, 2005). The listening tests were composed of two parts. In the first part, subjects were asked to rate how good each stimulus they heard was on a scale from 1 to 5 (worst to best). For the second part, subjects were required to listen to a set of stimuli and then type-in the sentence they heard.

Fig. 3 summarises the results of the first *Blizzard* Challenge evaluation. It was decided that the Blizzard Challenge organisers would not disclose the identities of the entrants, in order to encourage entries from the commercial

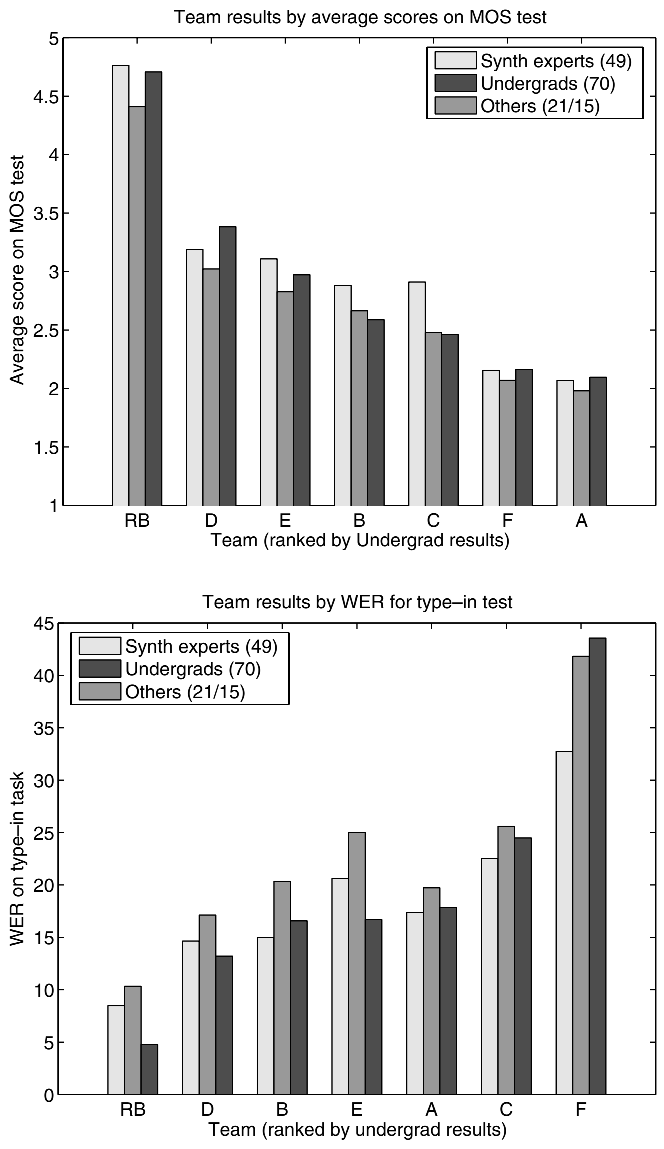

Fig. 3. Team results for the first *Blizzard* Synthesis Challenge competition. The top graph shows results for the Mean Opinion Score test, on a scale of 1–5, while the bottom plot shows results in terms of Word Error Rate in the type-in comprehension test. The *Blizzard* Challenge was made anonymous to encourage entries from the commercial sector, hence teams are identified by letter only. The Multisyn entry is team B. "RB" is real human speech.

sector, so the competing systems are known only by a letter. The team labelled “RB” in this figure in fact gives the results for real human speech. Meanwhile, team “D” has willingly identified itself as the Trajectory HMM entry from the Nagoya Institute of Technology, Japan. This was the only non-concatenative system in the competition, and is widely regarded as having benefited from the rather restricted size of the speech datasets used for this first *Blizzard* Challenge.

Out of the five concatenative systems in the challenge, Fig. 3 shows that the Multisyn entry, team B, came first in terms of WER and second in terms of MOS, as measured by taking the overall averages of these scores and normalising by the number of subjects in each group. This is a satisfactory result, especially since all four of the voices submitted were the product of purely automatic processing with no manual intervention or fixing of errors.

7. Summary and conclusions

The Multisyn engine works well and easily achieves its main goal of providing a good unit selection speech synthesis engine and the necessary tools to build new voices with limited speech synthesis knowledge and minimal effort. Experience with the system suggests that a corpus of around 36,000 phones provides an intelligible voice, but the system performs much better with much bigger corpora, the quality of the resulting voice being highly dependent on the quality of the database.

There is room for improvement in a few areas. The system as described here contains no real control over prosody, so intonation and duration of segments is not modelled explicitly; instead the context from which units are selected provides an implicit model of prosody. This works well most of the time, but there are occasions where the resulting speech has primary phrasal stress placed inappropriately. Ways of modelling prosody in the system are currently under investigation. One potential solution is to model primary phrasal stress alone (rather than full prosodic modelling). Along with a suitable accent prediction utility, this would still to allow the rapid, automatic building of voices.

Pronunciation variation is still a problematic area, and a way of addressing the issue throughout the system in a consistent manner – both during voice building and at synthesis time – is needed.

Overall, the simplicity of the Multisyn approach provides a robust and easy to use unit selection engine which is flexible and configurable, yet requires only a little expertise from the person building the voice.

Acknowledgements

This work was funded in part by EPSRC grant GR/R94688/01. King is supported by an EPSRC Advanced Research Fellowship.

References

Bennett, C.L., 2005. Large scale evaluation of corpus-based synthesizers: results and lessons from the Blizzard challenge 2005. In: Proceedings of the Interspeech 2005, Lisbon, pp. 105–108.

Bennett, C.L., Black, A.W., 2005. Prediction of pronunciation variations for speech synthesis: a data-driven approach. In: ICASSP 2005, Philadelphia, USA, Vol. I, pp. 297–300.

Beutnagel, M., Conkie, A., 1999. Interaction of units in a unit selection database. In: European Conf. on Speech Communication and Technology, Vol. 3. pp. 1063–1066.

Black, A., Lenzo, K., 2000. Limited domain synthesis. In: Proceedings of the ICSLP2000, Beijing, China.

Black, A., Lenzo, K., 2004. Multilingual text-to-speech synthesis. In: Proceedings of the ICASSP 2004, Montreal, Canada.

Black, A., Taylor, P., 1997. Automatically clustering similar units for unit selection in speech synthesis. In: Eurospeech’97, Vol. 2, pp. 601–604.

Black, A.W., Campbell, N., 1995. Optimising selection of units from speech databases for concatenative synthesis. In: Proceedings of the Eurospeech’95, Madrid, Spain, pp. 581–584.

Black, A.W., Lenzo, K.A., 2001. Optimal data selection for unit selection synthesis. In: 4th ISCA Workshop on Speech Synthesis, pp. 63–67.

Black, A.W., Tokuda, K., 2005. Evaluating corpus-based speech synthesis on common datasets. In: Proceedings of the Interspeech 2005, Lisbon, pp. 77–80.

Bozkurt, B., Ozturk, O., Dutoit, T., 2003. Text design for TTS speech corpus building using a modified greedy algorithm. In: Proceedings of the Eurospeech’03, Geneva, Switzerland, pp. 277–280.

Bulyko, I., Ostendorf, M., 2001. Joint prosody prediction and unit selection for concatenative speech synthesis.

Conkie, A., 1999. A robust unit selection system for speech synthesis.

Conkie, A., Isard, S.D., 1996. Optimal coupling of diphones. In: Santen, J.P.H., Sproat, R.W., Olive, J.P., Hirschberg, J. (Eds.), Progress in Speech Synthesis. Springer, Berlin.

Fitt, S., Isard, S., 1999. Synthesis of regional English using a keyword lexicon. In: Proceedings of the Eurospeech’99, Budapest, Vol. 2, pp. 823–826.

Foster, M.E., White, M., Setzer, A., Catizone, R., 2005. Multimodal generation in the COMIC dialogue system. In: Proceedings of the ACL 2005 Demo Session.

Garofolo, J.S., 1988. Getting started with the DARPA TIMIT CD-ROM: an acoustic phonetic continuous speech database. National Institute of Standards and Technology (NIST), Gaithersburgh, MD.

Hamza, W., Bakis, R., Eide, E., 2004. Reconciling pronunciation differences between the front-end and back-end in the IBM speech synthesis system. In: ICSLP 2004, Jeju Island, South Korea.

Hofer, G., Richmond, K., Clark, R., 2005. Informed blending of databases for emotional speech synthesis. In: Proceedings of the Interspeech, September 2005.

Hunt, A., Black, A., 1996. Unit selection in a concatenative speech synthesis system using a large speech database. In: Proceedings of the ICASSP 1996, Atlanta, USA, Vol. 1, pp. 373–376.

Kominek, J., Black, A., 2004. The CMU ARCTIC speech databases. In: 5th ISCA Speech Synthesis Workshop, Pittsburgh, PA, pp. 223–224.

Kurtić, E., 2004. Polyglot voice design for unit selection speech synthesis. Master’s thesis, University of Edinburgh.

Makashay, M., Wightman, C., Syrdal, A., Conkie, A., 2000. Perceptual evaluation of automatic segmentation in text-to-speech synthesis. In: Proceedings of the ICSLP 2000, Beijing, China.

Möbius, B., 2001. Rare events and closed domains: two delicate concepts in speech synthesis. In: 4th ISCA Workshop on Speech Synthesis, pp. 41–46.

Syrdal, A.K., Wightman, C., Conkie, A., Stylianou, Y., Beutnagel, M., Schroeter, J., Strom, V., Lee, K.S., Makashay, M.J., 2000. Corpus-based techniques in the AT&T NextGen synthesis system. In: Proceedings of the ICSLP 2000, Vol. 3, pp. 410–415.

Taylor, P., 2000. Concept-to-speech by phonological structure matching. Philosophical Transactions of the Royal Society Series A.

Taylor, P., Black, A., Caley, R., 1998. The architecture of the Festival speech synthesis system. In: Proceedings of the The Third ESCA Workshop in Speech Synthesis, pp. 147–151.

van Santen, J., Buchsbaum, A., 1997. Methods for optimal text selection. In: Eurospeech97, Vol. 2, pp. 553–556.

van Santen, J.P.H., September 1997. Combinatorial issues in text-to-speech synthesis. In: Proceedings of the Eurospeech, Rhodes, Greece, Vol. 5, pp. 2511–2514.

Vepa, J., King, S., 2004. Join cost for unit selection speech synthesis. In: Alwan, A., Narayanan, S. (Eds.), Speech Synthesis. Prentice-Hall, Englewood Cliffs, NJ.

Wells, J.C., 1982. Accents of English. Cambridge University Press, Cambridge, UK.

Wrench, A.A., 2001. A new resource for production modelling in speech technology. In: Proceedings of the Workshop on Innovations in Speech Processing.

Young, S., Evermann, G., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., Woodland, P., 2002. The HTK Book (for HTK version 3.2). Cambridge University Engineering Department.