DIPHONE COLLECTION AND SYNTHESIS

Kevin A. Lenzo[1] *, Alan W Black*[2]

[1]International Software Research Institute, [2]Language Technologies Institute,
Carnegie Mellon University,
{lenzo,awb }@cs.cmu. edu

ABSTRACT

In this paper, we describe the design and collection of corpora for diphone synthesis, the voice building process, and our experience in the creation of a new, publically available database of ten diphone sets of one American English speaker for the Festival Speech Synthesis System [3], using the FestVox document and tools [1]. In support of our goal to make the tools and techniques available for anyone to build their own synthetic voices, we have generalized and streamlined the tasks involved from what were once arcane anecdotes, half-written one-off scripts, and partial descriptions, to detailed, complete instructions that others have followed with good results.

1. INTRODUCTION

The FestVox [1] document is a growing, publically available resource that contains tools, data, and text about building complete synthetic voices in English and other languages. That work covers everything from building text analyzers, lexicons, prosodic models as well as various waveform synthesis techniques including general unit selection and diphones. Inside this framework, we have collected a considerable amount of data, and refined the process. While building good, characteristic voices cannot be reduced to a simple recipe, we hope this will be a starting point for those interested in speech science and technology, and will provide a common basis for the comparison of various diphone synthesis techniques over the same data sets.

Although diphone-based synthesizers are only one of a number of techniques documented in FestVox, we believe they are, at present, the most reliable and resource-effective method for building new voices for general text-to-speech synthesizers. A diphone here is two connected half-phones, where a "phone" here may in fact be any segment including a traditional phoneme, allophone or consonant cluster. We carefully construct examples of each phone-phone transition in our phoneset, so as to capture all the implied sequential articulatory transitions, even though some may not be phonotactically valid (like [ZH-NG]).

To fully exercise our techniques, we are collecting ten sets of diphones, at 32KHz, with simultaneous electroglottograph (EGG) signal, from a single speaker of American English (KAL) with varying speaking rate. These databases are being released publicly with an open license, so that anyone who wishes can replicate our findings, study the voice, teach about synthesis, or build their own by comparison. We, and others, have also used these techniques on other voices and other languages. Four sets have been recorded so far, and the bulk of this paper relates to our experience with that set and the tools that have been created to help – including a recording session management tool called *pointyclicky*.

2. DIPHONE SYNTHESIS

Diphone synthesis is one of the most popular methods used for creating a synthetic voice from recordings or samples of a particular person; it can capture a good deal of the acoustic quality of an individual, within some limits. The rationale for using a diphone, which is two adjacent half-phones, is that the "center" of a phonetic realization is the most stable region, whereas the transition from one "segment" to another contains the most interesting phenomena, and thus the hardest to model. The diphone, then, cuts the units at the points of relative stability, rather than at the volatile phone-phone transition, where so-called coarticulatory effects appear.

There is clearly a simplifying assumption: that all relevant phonetic realizations can be enumerated, and that by simply collecting all of phone-phone transitions, that any possible sequence of speech sounds in the target language could be produced. Thus, with a 44-phone inventory, one could collect a 44 * 44 = 1936 diphone inventory and create a synthesizer that could speak anything, given the imposition of appropriate prosody – intonation, duration, and shift in spectral quality, as determined by other modules in a general-purpose synthesizer.

3. COLLECTING A DIPHONE SET

Building a diphone synthesizer involves several steps:

3.1. Choosing the phone set

One should consider the phone set carefully for synthesis. For American English, we have been using a phone set based on the one we also use in the CMU Sphinx open source speech recognition system, which is a DARPAbet-style set of 44 phones. Languages with fewer phonemes can get away with fewer allophones, but we find this set adequate.

3.2. Designing carrier material

We use nonsense carrier words to collect all possible diphones, following [5]. Others have successfully used natural carrier phrases, but the argued advantage of natural delivery offers may also be a disadvantage as people may assume too much, and fail to produce exactly the desired phones. Within this framework, an experiment may be carried out to compare the results of voices made from nonsense words to one made from naturalistic text, but we know of none having been performed as of yet.

It should be noted that delivering diphones is not a particularly natural endeavor. As these phone segments will be extracted both for pitch and duration, it is important that their delivery be consistent, so that joins are more likely to be acceptable.

We believe that the use of nonsense carrier material makes the delivery of the diphones more consistent. Also, the pronunciation of a phonetic string is more clearly defined in these nonsense words than in elicited natural words. We generate carrier phrases so that, where possible, we can extract the diphones from the middle of a word. As it takes time for the human articulation system to start, we do not want to extract diphones from syllables at the start or end of words, unless these transitions to or into silence (SIL) are part of the diphone in question.

for example, from

SIL T AA B AA B AA SIL

we would extract the diphones [B-AA] and [AA-B], and from

SIL T AA T EY AE T AA SIL

we would get [EY-AE], as the [T-EY] and [AE-T] are taken from elsewhere, though one could indeed get all three from the one prompt.

For each class in the language, consonant-vowel, vowel-consonant, vowel-vowel, consonant-consonant, silence-phone, phone-silence, and any other special sets like syllabic, consonant clusters and allophones we build simple carrier sets and loop through all possible values generating a long list of strings of phones which contain all possible diphones in the language to be recorded. Basic scripts are provide for this that can be adapted for other language, while specific scripts are provided for currently supported languages.

3.3. Generating the prompts

Once the phone set and the sort of carrier material for each type has been chosen, the prompt list is produced automatically from the specification using a set of templates. Consonant-vowel and vowel-consonant pairs are generally kept together, as shown in the example above – [B-AA] and [AA-B] are generated in a single prompt. Special contexts are also created for e.g. vowel-vowel diphones, also as shown above, and for transitions to and from silences into a phone.

We then synthesize these prompts using an existing synthesizer. The prompts are generated for a number of reasons: first, to play to the user while recording. Even highly trained phoneticians make mistakes in reading 2000 prompts so the synthesized prompts help guide the speaker to say the right thing. Secondly, as we generate these prompts at constant pitch and duration, this encourages the speaker to do likewise. As we are going to modify the pitch and duration independently, it is better if the recording is in a monotone, and consistent. This, we feel, is best done by effectively having the nonsense word delivered in what almost sounds like a chant.

The second reason for generating prompts is their use in labelling the spoken word which is described below.

3.4. Preparing the audio collection environment

Ideally, speech data is collected in an anechoic chamber, with high-quality recording equipment, in a comfortable setting, at CD-quality, with simultaneous audio and electroglottograph (EGG) signal. In practice, we have collected KAL1 in a soundproof booth in the Electrical and Computer Engineering lab at CMU, and then collected KAL2 through KAL4 at one of the author's apartment, largely between 4 and 6 AM, before traffic got started.

We used a Shure SM-2 close-talking headset microphone, a Symmetrics SX202 microphone preamp, a Glottal Enterprises EG2-PC electroglottograph, and a SoundBlaster X-Gamer PCI audio card, with various recording and playing utilities on a decent machine running Linux. The computer was on an uninterruptible power supply (UPS), which reduced electrical noise. We used a wireless keyboard and mouse, so that the subject could sit back several feet from the computer monitor – which otherwise introduced considerable noise into the recordings. The radio frequency from the wireless keyboard and mouse appeared to have no detrimental effect.

In general, we record diphone sets at 32 KHz, with simultaneous audio and EGG signal, after making sure the levels are sane for both (sane being peaking in the 80% range). After collection, however, these signals are split into separate files.

We have also collected diphone sets directly to a laptop computer in a quiet room (i.e. one without air-conditioning or other computers, which isn't easy on the CMU campus). Laptops should be run on battery power to reduce hum. The audio systems on some laptops, however, are not good high enough quality for recording, and ensuring that the machine's audio device is good enough is very important. For other synthesis techniques, such as the limited domain synthesizers [2] we have built in the FestVox framework, the audio quality is less important as there are typical multiple examples of each phone type. In our diphone database there are often only one examples and hence *every* part of the recording *must* be good.

3.5. Collecting the data

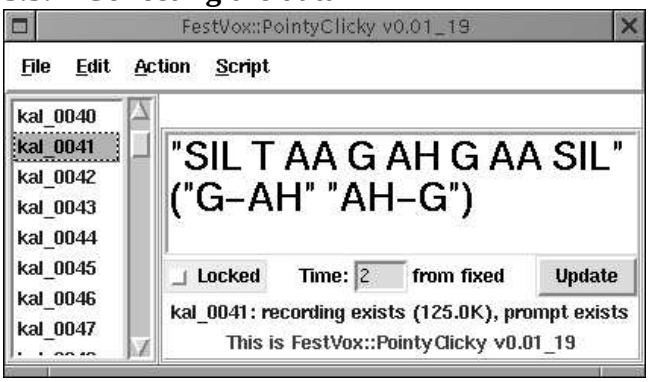

Figure 1: pointyclicky

For data collection, we use either an application called "pointyclicky" (Figure 1) written in Perl with Perl/Tk, or a simple

shell script called "prompt_them;" both are in the FestVox distribution. With pointyclicky, the management of the recording session is simplified. You can double-click with the left mouse button to play the prompt (or the recording, if it exists), as well as make a selection and act over it – such as iterating over the selected files with the sequence of displaying the text, playing the prompt, pausing briefly, and then recording.

To aid in quickly visualizing what prompts have and have not yet been recorded, the color of the item's name, as well as the background in the scrolling list, changes based on state: if the prompt does not exist, the background of the cell is gray; if it does, it is white. If the recorded file exists, the text is blue; if not, it appears in red. This allows the user to quickly navigate over large lists and see what is and is not finished. Bad "takes" can be easily deleted, and their status is immediately reflected in the list.

The two menus, "Action" and "Script," are user-extensible. The program looks into a hidden directory that contains two subdirectories of the same names, and these contain the various Perl programs that are executed when these are selected in from the menu. Items in the "Action" menu iterate over the items selected in the scrolling list on the left of the user interface in pointyclicky, whereas those in the "Script" menu act independently of the selection, to do things such as instantiate an example talking clock example, or invoke festival to perform the processing to complete the voice building.

One handy feature of pointyclicky is that, despite its name, one needn't use a mouse. All the functions are accessible as simple keyboard shortcuts, which allows the voice talent to cruise around over the files easily, and hear what is recorded and the prompts. When one wants to start recording whatever remains to be done, one simply uses the Edit menu to select the remainder, then selects the appropriate action – the default being to display the prompt, play the synthesized prompt, turn the background color to yellow, pause for 0.75 seconds, turn the background to red, record for the specified time, and then turn the background back to the default color. All of these commands have been used to extend the Perl language, and are documented in the FestVox Perl module available at `http://festvox.org/pointyclicky` . Other actions are available by default too, and the user can simply drop Perl programs into the appropriate directories, or edit the Preferences for pointyclicky's behaviour in a dialog box.

Anecdotally, we have found that this interface helps both speed up the recording process, and reduce the number of errors. It certainly makes the recording process more pleasant and manageable.

3.6. Segmentation/Alignment

We use a simple, effective technique based on [6] to segment the recorded prompts, by using DTW to align the pre-generated, labeled synthesized diphone prompts to the new recordings. This is the same technique we use for aligning data for limited domain synthesis; the technique has worked quite well for us to make initial passes, even when the source language and the target language differ: American English has been used to generate first-pass alignments for both Croatian and Nepali.

To confirm this technique's accuracy we used it against some earlier databases which had been hand labelled.

	type	RMSE	stddev
KED-KED	self	14.77ms	17.08
MWM-KED	US-US	27.23ms	28.95
GSW-KED	UK-US	25.25ms	23.92
KED-WHY	US-Kor	28.34ms	27.52

KED is a US English voice, collected at University of Edinburgh, MWM was collected at Oregon Graduate Institute. A KED voice built directly from this fully automatic labelling technique was certainly understandable though not as good as the hand labelled form. GSW is a British English voice yet we used this to label US English (with reasonable mappings for phone names). Again the results were reasonable. The last example used KED, the US English voice, to align against WHY a Korean voice. In spite of these being different languages, the results was perfectly usable.

Even though there is a phonetic mismatch between English and Korean – English uses aspiration of stops in free variation as alloophones, whereas Korean has a phonemic distinction between aspirated and unaspirated stops – we note that mostly labelling is correct but for a small number of labels they are completely wrong having aligned to some lip smack, or some other noise or artifact. The success of this should not be surprising. This is a very constrained labelling task. We know *exactly* what phones are present and hence alignment should be trivial. In fact if it is not trivial it is likely there is a problem with the recording.

We do not pretend that this is perfect, but the alignments are very good as a first pass. We do usually also hand check all labelling and move boundaries as is required to get the best performance. But this level of fine tuning was also done in the days that we relied solely of hand labelling. In those days, initial hand labelling was always somewhat rough and prone to error. This technique has allowed us to remove that stage producing initial results in minutes rather than days.

3.7. Quality control

After the automatic alignment and labeling (often called "autolabeling") is finished, the results are visually inspected by either random sample, or by checking every one, using display tools such as *emulabel* [4]. We also run a number of scripts that check the durations and other features of each segment, to find those that are quite obviously out automatically, so as to go in with emulabel and correct them by hand.

In collecting these diphone sets we have noted that the quality of autolabelling has improved as we typically use the previous set as the prompts. Of course this is the same voice delivered (mostly) in the same style thus later versions have not required full checking and sampling and targeting problems alone has been sufficient.

3.8. Extracting pitchmarks and pitch-synchronous parameters

Once the recordings are made and are labelled, building the diphone synthesizer itself is mostly automatic. Our first stage is to

extract the pitchmarks from the EGG signal, if an electroglottograph was used at recording time. We use the *pitchmark* program that is part of the Edinburgh Speech Tools, on which Festival is built, to do this. For diphone databases for which no EGG is provided, we can extract the pitchmarks form the waveform files directly, but this is typically not as good from the EGG signal. For all voice sections of the speech, we position the pitchmarks at the peak of the pitch period. For non-voiced sections, we introduce a "fake" or pitch mark evenly spaced through those sections. As our signal techniques for pitch and duration modification depend of pitch synchronous analysis, getting the pitch marks right is very important to the final quality of the synthesizer.

Although we try hard to ensure that the audio quality remains constant throughout the whole recording, it is unusual for the whole set to be done, perfectly, in a single sitting, and we have found that slight differences in power occur between different sittings, due to position of the microphone as well as the speaker delivering with different vocal effort. To combat this, we include a simple power normalization phase. As different phones have different inherent power can cannot simply normalize everything; therefore, we calculate the mean RMS power over all vowels in each nonsense word, then find the mean over all the files, and calculate a modification factor for each word that in the normalization.

After power normalization, we extract LPC parameters pitch synchronously.

3.9. Diphone index

As diphones run from mid of one phone to mid of another, we need to know exactly where that "mid" is. For supported languages, we already know where the diphone boundary is in existing diphone databases, so when we synthesize the prompts, the accompanying labels include both the phone boundary positions, as well as the diphone boundaries. Although midway between phone boundaries may be the most appropriate join point for vowels, it almost certainly is not for stops, where the closure part of the phone is by far a better place to join. Diphone boundaries (marked as "DB") are also often the part requiring correction.

From the labels, we build a diphone index automatically, which can be used by Festival to synthesize waveforms. Two basic methods are offered first: so-called "separate-mode," where the diphones are selected from each LPC and residual file on demand, and "group-mode," where we can collect just the diphone parts and put them into a single large file. The first of this is used in the initial debugging stage. The second stage is used for distribution of complete voices, as it is both more compact and quicker to access.

3.10. Amalgamation

A final voice consist of not just a diphone set, but also the front end of the TTS system including text analysis, lexicons and prosodic models. These, in contrast, although difficult in themselves, are much smaller that the diphone set, and for English at least are a standard part of our distribution.

Thus, the building a new English voice can simply be a matter of recording a new diphone set. Our tools provide complete scripts and detailed walk-through for this process building on top of the existing modules.

4. CONCLUSIONS

In building so many diphone database, and particularly by repeating the process a number of times with the same speaker in the same dialect, we feel we have streamlined our build process so that it is much more reliable. In other systems, building new voices is such an undertaking that it is not something that can easily be experimented with, without significantly more work. We can record, label and hand-check a complete US English diphone set in a day, albeit a long day, and with only preliminary quality; but, with such turnaround rate, we have be able to identify specific problems that we have had to address.

Even when the speaker is an expert in phonetics and diphone synthesis, we know it is still very easy to make phonetic mistakes in recording. The vowel-vowel transitions are notably difficult to produce. They are relatively rare in normal speech but of course as we are collecting complete coverage we need instances of *all* examples. We have also noted that the phones [AX] and [AH] are particularly difficult to reliably produce, even when we are keenly aware of the trouble spot.

The US English databases themselves are available from `http://festvox.org/dbs/index.html`, and the full documentation with scripts, code and explicit walk-troughs of these techniques with examples are available at `http://festvox.org`.

5. REFERENCES

1. Black, A., and Lenzo, K. Building voices in the Festival speech synthesis system. http://festvox.org, 2000.
2. Black, A., and Lenzo, K. Limited domain synthesis. In *ICSLP200* (Beijing, China., 2000).
3. Black, A., Taylor, P., and Caley, R. The Festival speech synthesis system. http://www.cstr.ed.ac.uk/projects/festival.html, 1998.
4. Cassidy, S. The EMU speech database system. http://www.shlrc.mq.edu.au/emu/, 2000.
5. Isard, S., and Miller, D. Diphone synthesis techniques. In *Proceedings of IEE International Conference on Speech Input/Output* (1986), pp. 77–82.
6. Malfrere, F., and Dutoit, T. High quality speech synthesis for phonetic speech segmentation. In *Eurospeech97* (Rhodes, Greece, 1997), pp. 2631–2634.