Automatic Close Copy Speech Synthesis

Synteza mowy metodą automatycznego dokładnego kopiowania

Jolanta Bachan

Adam Mickiewicz University, Institute of Linguistics

jolabachan@gmail.com

ABSTRACT

The aim of the present study is, first, to develop a restricted domain speech synthesis concept for automatically generating acoustic stimuli for use in evaluating cochlear implants for children and, second, to implement a prototype synthesiser. The approach taken is to use Close Copy Speech (CCS) synthesis whose task it to "repeat utterances produced by a human speaker with a synthetic voice, while keeping the original prosody" [1]. The paper concentrates on a sub-domain of CCS, namely on Automatic Close Copy Speech (ACCS) synthesis, in which the transfer of parameters from the original speech signal and annotation is performed automatically. The design and implementation of ACCS is presented.

STRESZCZENIE

Celem obecnego badania jest, po pierwsze, opracowanie konceptu syntezy mowy z ograniczonej dziedziny dla automatycznego generowania bodźców akustycznych na potrzeby oceny implantów ślimakowych dla dzieci i, po drugie, implementacja prototypowego syntezatora. Jako metodę przyjęto syntezę mowy metodą dokładnego kopiowania (Close Copy Speech (CCS) synthesis), której zadaniem jest „powtarzanie wypowiedzi produkowanych przez ludzkiego mówcę za pomocą głosu syntetycznego, z zachowaniem oryginalnej prozodii" [1]. Artykuł ten koncentruje się na poddomenie CCS, a mianowicie na syntezie mowy metodą automatycznego dokładnego kopiowania (Automatic Close Copy Speech (ACCS) synthesis), w której przeniesienie parametrów z oryginalnego sygnału mowy i anotacji odbywa się automatycznie. W pracy zaprezentowany jest projekt i implementacja ACCS.

1. Introduction

The aim of the present study is, first, to develop a restricted domain speech synthesis concept for automatically generating acoustic stimuli for use in evaluating cochlear implants for children and, second, to implement a prototype synthesiser. The main motivation for including a speech synthesiser in the system is to increase the flexibility of the available test stimuli.

In the present form of the evaluation, tests of cochlear implants recorded stimuli are used. But a recorded custom corpus of recordings is static and inflexible. A possible solution is to use speech synthesis, which is dynamic and flexible. Moreover, synthetic speech allows presentation of carefully controlled speech-like stimuli to listeners in order to obtain judgements on their speech perception. The approach taken is to use Close Copy Speech (CCS) synthesis as a best case example of synthetic speech.

2. Close Copy Speech (CCS) synthesis

Close Copy Speech (CCS) synthesis or resynthesis method "repeats utterances produced by a human speaker with a synthetic voice, while keeping the original prosody" [1]. In this method, "close copy" means that the synthetic speech is as similar as possible to a human utterance.

In the present study, the definition by Dutoit is interpreted to mean that the Natural Language Processing or Text-To-Speech (TTS) component of the synthesiser is replaced by an analysis of a recorded speech signal. In the present context, "copy" therefore means that the input to the synthesis engine for a given utterance is derived directly from a corresponding utterance in the annotated corpus data. The system consists of a recorded speech signal, a method for pitch extraction from the speech signal, and a time-aligned phonemic annotation of the speech signal. The Close Copy procedure can be manual or automatic [2]; the present study reports on the development of Automatic Close Copy Speech (ACCS) synthesis.[1]

3. Requirements for ACCS synthesis

3.1. System requirements

The inputs required by the ACCS synthesis system are as follows:

1. Source speech (source speech recordings) and source DB (annotated speech database).
2. Speech synthesiser: diphone database (voice), synthesis engine.

The outputs to be produced by the ACCS synthesis system are as follows:

1. Target pronunciation specification: specification table for input to speech synthesis engine.
2. Target acoustic output: produced by the speech synthesis engine.

[1]The author would like to thank Dafydd Gibbon for advice on ACCS synthesis system development.

3.2. Recordings

A corpus of recordings[2] for a male voice was available from a speech synthesis development scenario. The texts were spoken by a professional speaker and the recordings were made in a professional recording studio. The sampling rate of the data in the available format is 16kHz in a standard WAV format. The texts for use in the synthesiser development consisted initially of a selection of 1200 sentences from the corpus of approximately 3200 utterances.

3.3. Annotations

Annotation of the recordings at phoneme level was performed automatically using the software tool CreatSeg [3] and checked by trained phoneticians. Phonemic segments which were not correctly handled by the automatic segmentator were manually edited. Additionally, the annotations also contain prosodic information, based partly on functional judgments and partly on prosodic information [2], [4].

3.4. Diphone database

The diphone database used in the study is the PL1 MBROLA Polish female diphone database[3] created under the free database access terms of the MBROLA project [5]. The diphone database consists of 1443 diphones and contains 37 phonemes in standard Polish SAMPA notation. No Polish male diphone database is available for MBROLA.

4. MBROLA diphone synthesiser architecture

For ACCS synthesis, the MBROLA diphone synthesis procedure was adopted, with modifications to the standard architecture, which has the following structure:

1. Natural Language Processing:
    1. Phonetisation: grapheme-to-phoneme conversion.
    2. Prosody generation: text parser for duration lookup and pitch assignment.
2. Pronunciation specification table (PHO file) as NLP-DSP (Natural Language Processing-Digital Signal Processsing) interface.
3. Speech synthesis component:
    1. Diphone database.
    2. MBROLA engine.
4. Audio (WAV file) output.

[2]The author gratefully acknowledges the provision of this corpus by Grażyna Demenko (Principal Investigator of the Cochlear Implant Evaluation project).

[3]Created by Krzysztof Szklanny and Krzysztof Marasek, whose work I gratefully acknowledge.

The NLP component is replaced by an annotation file in which a transcription and a time stamp are aligned with the speech signal recording. The annotation and the recording together in principle include all the information which is needed for generating the specification table interface to the synthesis engine, which is normally produced by the NLP component. Consequently, in ACCS synthesis no input text is used. ACCS synthesis makes use of recordings of real utterances and annotations derived from these recordings. In the annotation files, phonemes and their durations are stored. Information about pitch in relation to the phonemes in the annotation files is extracted from the speech recordings.

5. What is the NLP-DSP interface?

The central component for present purposes is the NLP-DSP interface which contains the pronunciation specification table produced by a TTS or CCS component, and used as input by the synthesis engine to synthesise speech. In MBROLA the NLP-DSP interface is implemented by so-called PHO files. The format specifies a table with three columns [6]:

1. phonemes that are present in the sound to be produced,
2. duration of these phonemes,
3. pitch values represented by one or more pairs of numbers - the first number stands for the place of the pitch value in the phoneme, the second number is the pitch value itself.

The syntax of the specification table ST is defined as a sequence of one or more vectors SV, each with three components: the phoneme PH, the phoneme duration PD and the sequence of zero (for voiceless stretches) or more pitch pairs PP (in the prototype maximally one), consisting of pitch location PL and the pitch value PV:

```
<ST>   ::= <SV>+
<SV>   ::= <PH> <PD> <PP>*
<PP>   ::= <PL> <PV>
<PH>   ::= sampa_phoneme1 | ... | sampa_phonemen
<PD>   ::= millisecond_integer
<PL>   ::= pitch_location_percent
<PV>   ::= pitch_value_hertz
```

An illustration of the first five rows of the pronunciation specification table interface between the NLP and the DSP components is shown in Table 1; this example was derived from the corpus.

Table 1: Fragment of Specification Table (ST) for MBROLA PHO file.

PH phoneme (PL1 SAMPA)	*PD phoneme duration (msec)*	*PP pitch pair*	
		PL pitch location (%)	*PV pitch value (hertz)*
n	66	50	200
a	72	50	210
S	82	50	240
tS	45	50	310
e~	29	50	306

6. Automatic Close Copy Speech synthesis design

6.1. Basic principles of ACCS synthesis

Automatic Close Copy Speech (ACCS) synthesis with an MBROLA type diphone synthesis is a process of automatically creating pronunciation specification tables (NLP-DSP interfaces, implemented as PHO files), making use of recorded and annotated real utterances, and synthesising the pronunciation specification tables using an appropriate voice (diphone database). The voice may be created from the annotated utterances, in the ideal case, or may be an independently created voice, as in the case of the present study.

The ACCS procedure therefore emulates the Natural Language Processing front end to a speech synthesis engine. The speech and annotation information are transformed automatically into a pronunciation specification table which, together with a diphone database, constitutes the input to the synthesis engine, which converts the specification table into speech using the diphone database. The acoustic output is a speech file. Figure 1 shows the general architecture of the ACCS synthesis system [2].

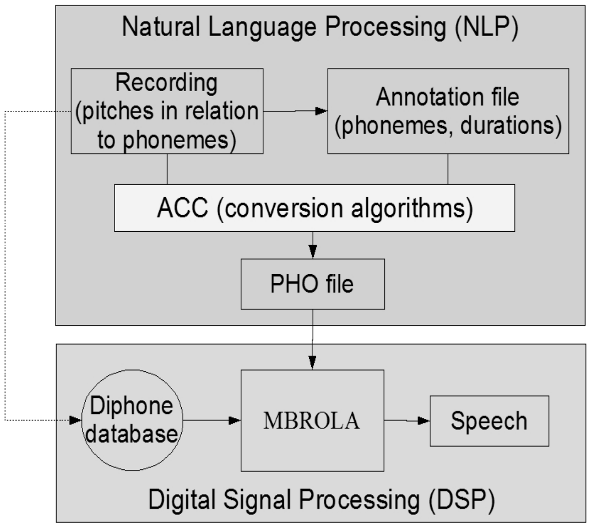

Natural Language Processing (NLP)
Recording (pitches in relation to phonemes)
Annotation file (phonemes, durations)
ACC (conversion algorithms)
PHO file
Diphone database
MBROLA
Speech
Digital Signal Processing (DSP)

Figure 1: The architecture of ACCS synthesis system.

6.2. Components of the ACCS synthesis system

ACCS synthesis is a complex process of converting an annotation file into a sound file while preserving the pitch pattern which occurs in the speech file for which the annotation file is made. The components of the ACCS synthesis system are:

1. Speech information input:
    1. speech recordings,
    2. time-aligned annotations of speech recordings in BLF format.
2. Speech synthesiser:
    1. diphone database,
    2. synthesis engine.
3. Pitch extraction script:
    1. BLF to MBROLA PHO format conversion procedure,
    2. MBROLA to TextGrid (Praat format) procedure,
    3. pitch extraction (calling Praat script),
    4. inclusion of pitch values in MBROLA PHO file,
    5. synthesis of PHO file with MBROLA engine.

6.3. Mismatches and format preprocessing

The specification table required by the speech synthesis engine when used with the available Polish diphone database resource differs from the table provided by the annotated speech resource. This incompatibility has several components, for which format conversion tools need to be specified. The incompatibilities are listed in Table 2.

Table 2: Polish annotation, diphone database and pronunciation specification table (NLP-DSP interface) conventions

Polish annotation	*Diphone database*	*NLP-DSP interface*
sample numbers	-	durations (msec)
positional allophones	phonemes	phonemes
BLF phoneme set	PL1 phoneme set	PL1 phoneme set
syllable boundaries	-	-
word boundary types	-	-
pauses	pauses	pauses
prosodic annotation	-	-

The missing boundaries and the stress markings are not usable in the current configuration and are deleted, but will be considered at a later stage for prosody parametrisation. The SAMPA phoneme set and notation was given by the available diphone database in the pre-processed input format, and differs from the phoneme set used in the corpus annotation. The correspondences are shown in Table 3.

Table 3: Mismatches between BLF and PL1 SAMPA.

BLF SAMPA annotation labels	*PL1 SAMPA symbols*	*BLF SAMPA annotation labels*	*PL1 SAMPA symbols*
p	p	i	i
b	b	y	I
t	t	e	e
d	d	a	a
k	k	o	o
g	g	u	u
c	-	@ - *English schwa*	-
J	-	-	e~
f	f	-	o~
v	v	m	m
s	s	n	n
z	z	n'	n'
S	S	N	N
Z	Z	l	l
s'	s'	r	r

BLF SAMPA annotation labels	*PL1 SAMPA symbols*
z'	z'
x	x
t^s	ts
d^z	dz
t^S	tS
d^Z	dZ
t^s'	ts'
d^z'	dz'

BLF SAMPA annotation labels	*PL1 SAMPA symbols*
w	w
j	j
w~	-
j~	-

A further mismatch occurs between the Polish annotation interface (BLF) and the NLP-DSP interface (PHO) formats for time specification. The BLF format includes sample numbers, while the PHO format requires durations. In order to calculate durations, sampling rate metadata information (16 kHz) is required. The formula for bridging the gap is $(samplenumber_i - samplenumber_{i-1}) / samplingrate$.

Perhaps the most crucial mismatch is between the corpus, which is recorded using a male voice, and the diphone database, which is derived from a female voice. This requires a pitch re-adjustment. Currently the trivial formula $pitch_{female} = 2 * pitch_{male}$ is used, but parametrisations with more complex formulae incorporating a baseline are being developed. In the long term, an annotated corpus based on a female voice is required, as well as diphone databases based on male voices.

6.4. ACCS synthesis development procedure overview

The ACCS synthesis requires several conversion steps. The overall implementation architecture is shown in Figure 2. Since the Praat script for extracting the pitch values requires a different format (TextGrid), the BLF sample number and phoneme notation were converted into both MBROLA PHO format and Praat TextGrid format. The conversion was performed from the PHO format into TextGrid format, i.e. indirect conversion of BLF into TextGrid format, because BLF had already been converted into PHO format.

Figure 2 presents the overall ACCS synthesis implementation. The explanation of the figure is to be found in the following sections.

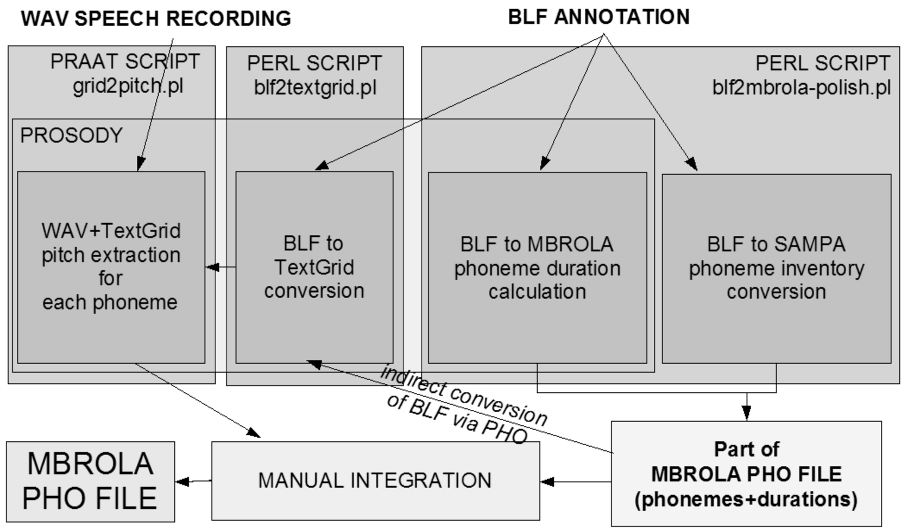

WAV SPEECH RECORDING
BLF ANNOTATION
PRAAT SCRIPT grid2pitch.pl
PERL SCRIPT blf2textgrid.pl
PERL SCRIPT blf2mbrola-polish.pl
PROSODY
WAV+TextGrid pitch extraction for each phoneme
BLF to TextGrid conversion
BLF to MBROLA phoneme duration calculation
BLF to SAMPA phoneme inventory conversion
indirect conversion of BLF via PHO
MBROLA PHO FILE
MANUAL INTEGRATION
Part of MBROLA PHO FILE (phonemes+durations)

Figure 2: Detailed schema of Automatic Close Conversion Speech synthesis (manual integration was later automnatised).

Figure 3 shows the detailed modular structure of the ACCS synthesis system, which is currently implemented as Perl and Praat scripts.

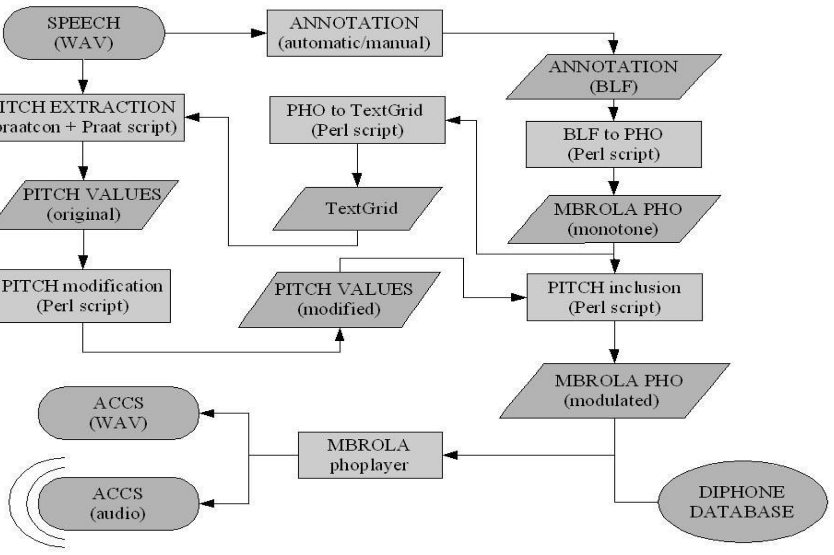


Figure 3: The architecture of the ACCS system.

6.4.1. Automatic BLF phoneme set to Polish Female Voice (the diphone database) phoneme set conversion

The main problem which appeared in the process of synthesising speech using the Close Copy Speech synthesis method was that phoneme set used for annotating recorded utterances (BLF SAMPA annotation phoneme set) was not the same as the phoneme set used by the diphone database for the Polish language (PL1 SAMPA phoneme set). Mapping most of the BLF annotation labels used by the Polish Female Voice (the diphone database) was not difficult. However, in some cases adapting the phoneme set used for annotation to the diphone database phoneme set was very tricky. The problem was caused by [ew~], [ow~], [ej~] and [oj~] sequences of phonemes in the BLF inventory, because those phonemes were equivalent to [e~] and [o~] present in the diphone database inventory. The sequences of phonemes [e] or [o] followed by [w~] or [j~] must be replaced by one segment [e~] or [o~], depending on the context. Additionally, the duration of the [e~] or [o~] phoneme was a sum of the durations of [ew~], [ow~], [ej~] or [oj~].

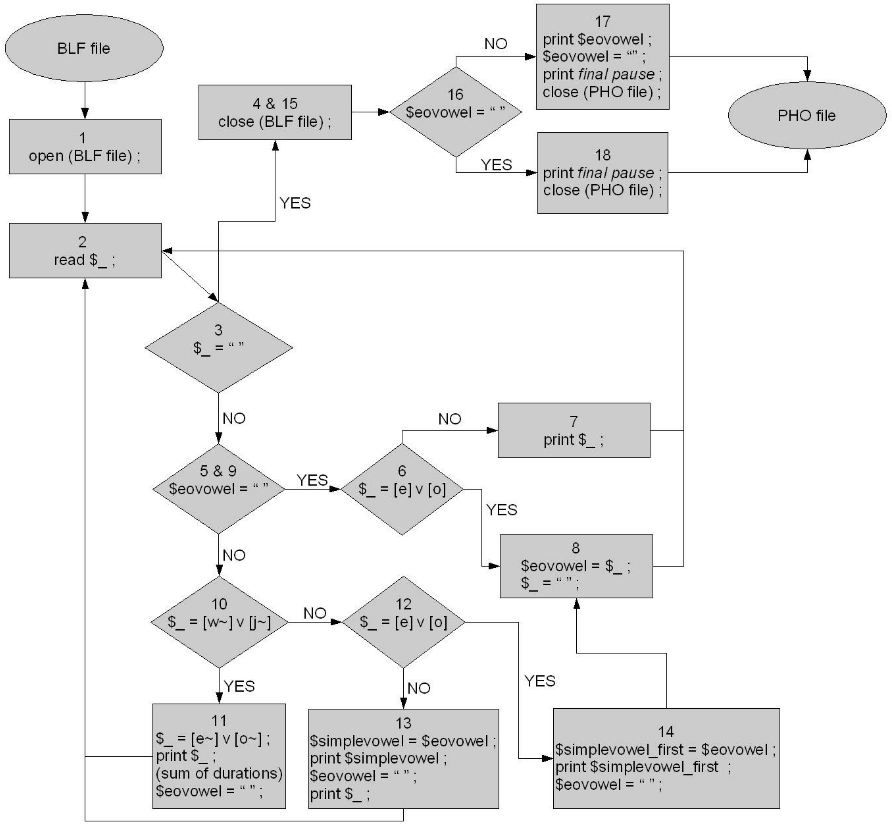

BLF file
1 open (BLF file) ;
2 read $_ ;
3 $_ = " "
4 & 15 close (BLF file) ;
16 $eovowel = " "
17 print $eovowel ; $eovowel = "" ; print final pause ; close (PHO file) ;
18 print final pause ; close (PHO file) ;
PHO file
5 & 9 $eovowel = " "
6 $_ = [e] v [o]
7 print $_ ;
8 $eovowel = $_ ; $_ = " " ;
10 $_ = [w~] v [j~]
11 $_ = [e~] v [o~] ; print $_ ; (sum of durations) $eovowel = " " ;
12 $_ = [e] v [o]
13 $simplevowel = $eovowel ; print $simplevowel ; $eovowel = " " ; print $_ ;
14 $simplevowel_first = $eovowel ; print $simplevowel_first ; $eovowel = " " ;
YES
NO

Figure 4: Schema for BLF to PHO conversion..

Figure 4 shows a flow chart for conversion of the BLF SAMPA annotation phoneme set into the PL1 SAMPA phoneme set and creating PHO format from a BLF format for single files. The input to the program is one BLF file. The program reads the first line of the file, makes the conversion, prints the converted line in PHO format into the PHO file. Then the program reads another line, makes the conversion, prints the converted line in PHO format into the PHO file, etc. When all the lines are read, converted and printed into the PHO file, the PHO file is closed and it can be later input to MBROLA for synthesising speech. A detailed description of the algorithm visualised in Figure 4 follows.

1. The program opens a BLF file.
2. The program reads the first line if the file is not empty. The phoneme from the line is put into the $_ variable.
3. The program checks if $_ is empty.
4. If the $_ variable is empty, it means that either the BLF file is empty or the program has already read all the lines. Then BLF file is closed.
5. If the line is not empty, then the program puts the values from the line into variables, and checks if the variable $eovowel is empty.
6. If the $eovariable is empty, then the program asks if the variable $_ is equal to the vowel [e] or equal to the vowel [o].
7. If $_ does not contain [e] or [o], then it prints the value of the $_ variable to the PHO file and reads another line from the BLF file.
8. If $_ contains [e] or [o], then it does not print anything, stores the value of the $_ variable in the $eovowel variable and reads another line from the BLF file.
9. Then a new line is read if there is another line, i.e. if all the lines have not been read already. If there is another line, then the program checks if the $eovowel variable is empty.
10. If $eovowel is not empty, then it asks if the $_ variable contains [w~] or [j~].
11. If the $_ contains [w~] or [j~], then [w~] or [j~] is converted into [e~] or [o~], the duration of the new (current) phoneme is calculated by adding the duration of the previous phoneme, $eovowel, and the current phoneme, $_. Then the value of the $_ variable is printed, i.e. [e~] or [o~]. The $eovowel is emptied and the program reads a new line from the BLF file.
12. If $eovowel is not empty, and the $_ variable does not contain [w~] or [j~], then the program asks if the value of the $_ variable is equal to [e] or [o].
13. If $_ is not equal to [e] or [o], then the program treats the previous vowel [e] or [o] (because $eovowel is not empty) as a simple vowel and stores the vowel in the $simplevowel variable. Then the value of the $simplevowel variable is printed, as well as

the phoneme stored in the $_ variable. The $eovowel variable is emptied and the program reads a new line from the BLF file.

14. If $eovowel is not empty, and the $_ variable is equal to [e] or [o], then the program knows that the first vowel stored in the $eovowel variable was a "simple vowel", puts the value of the $eovowel variable into a new variable called $simplevowel_first, prints the value of the $simplevowel_first, empties the $eovowel variable in order to put the value of the $_ variable in it. Now $eovowel is equal $_ ([e] or [o]). The current phoneme $_ is not printed (because it is [e] or [o]). The program reads a new line from the BLF file. The proces of examining which phoneme follows the phoneme [e] or [o] in the BLF file repeats, because the $eovowel is again not empty.

15. If there are no more lines in the BLF file, it means that the $_ variable is empty and all the lines have been read. The BLF file is closed.

16. Then the program checks if the $eovowel is empty. The $eovowel variable is not empty if [e] or [o] is the last phoneme in the BLF file. Because there is no phoneme which follows, the $eovowel has no chance to be printed while going through the "$eovowel procedure" - when $_ is not empty, and $eovowel is not empty. Therefore, the value of the $eovowel variable must be checked after having read all the lines in the BLF files and closing the BLF file.

17. If the $eovowel variable is not empty, then the value of the $eovowel is printed. It is followed by printing the reconstructed final pause. Because it is not possible to count the duration of the final pause, the final pause gets the value of 200msec. The PHO file is closed.

18. If the $eovowel variable is empty, then the reconstructed final pause is printed into the PHO file and the PHO file is closed.

6.4.2. Which problems connected with the phoneme set conversion are not solved by the program?

The program does not the deal with two Polish phonemes [c] and [J]. These phonemes are present in the BLF SAMPA annotation phoneme set, but are not included in the PL1 SAMPA phoneme set. These phonemes are reconstructed by the sequences of phonemes [kj] or [ki] for the phoneme [c] and by [gj] or [gi] for the phoneme [J]. The problem is illustrated in Table 4. The star "*" after the brackets means that the phonemes in the brackets may or may not appear after the phonemes [c] and [J].

Table 4: The phonemes [c] and [J] from the BLF SAMPA annotation convention and their equivalents in the PL1 diphone database.

BLF SAMPA annotation labels		*PL1 SAMPA symbols*	
c	(j/i)*	k	j/i
J	(j/i)*	g	j/i

For the available annotation files, the occurrence of [c] and [J] was transcribed as [c] followed by [j] or [i] and [J] followed by [j] or [i], creating sequences of phonemes [cj] and [Jj]. Having these sequences of phonemes, it is easy to replace the phoneme [c] by the phoneme [k] and the phoneme [J] by the phoneme [g] without any additional sequential split. If the [c] or [J] were not followed by [j] or [i], then there would have to be introduced a proces of splitting the phonemes [c] and [J] into sequences of [kj] or [ki] and [gj] or [gi], respectively. Then the duration of one phoneme would have to be split and one part of the value of the duration would have to be given to the phoneme [k] or [g] and the other part of the duration given to the phoneme [j] or [i].

It is hoped that all the other annotation files existing in the corpus transcribe the occurrence of [c] and [J] in the same way, i.e. [c] or [J] followed by the phoneme [j] or [i]. If there are cases that [c] and [J] are not followed by the phoneme [j] or [i], then the described above sequential split will have to be introduced.

To illustrate the problem, different transcriptions of the word “kiedy” is presented in Table 5:

Table 5: Different transcriptions of the word "kiedy."

BLF notation	*BLF notation with the sequential split procedure needed*	*PHO notation*
c	c	k
j		j
e	e	e
d	d	d
y	y	I

6.4.3. Automatic duration calculation

Calculation of the phoneme durations in milliseconds from the time stamps with information on the sampling rate was not difficult. But it has to be stated that in the BLF files the time stamps are marked at the beginning of the phonemes in the BLF files, therefore the duration of the last segment cannot be measured. The formula for calculating the duration of phonemes is

$$(samplenumber_i - samplenumber_{i-1})/samplingrate.$$

In other words, the sample number from the following phoneme is subtracted from the sample number on which the program operates currently and the result of the subtraction is then divided by the sampling rate. This means that before printing a phoneme from a line, the program must read another line to calculate the phoneme's duration. To solve the problem, a variable $_ (Perl standard line variable) which stores the value of the previous phoneme is introduced. To calculate the duration, the value of the duration of the previous phoneme must be stored. This problem is also solved by introducing a variable called sample_1. To be more explicit,

1. the program stores values of the previous line,
2. reads a following line,
3. does the conversion of the phoneme from the previous line,
4. calculates the duration of the phoneme from the previous line making use of the values of the sample number on the current line,
5. checks if the phoneme from the previous line meets the printing conditions,
6. if the phoneme from the previous line meets the printing conditions then this phoneme is printed; if it does not, then another line is read and the values from the penultimate and previous line are stored.

6.4.4. Praat pitch extraction

The extraction of pitch is the next step. Copying the phonemes, the durations of the phonemes from the annotation file and measuring the pitch values from the original recording of a human utterance allows best case speech synthesis.

To extract pitch from the recordings a Praat script called *max_pitch* was implemented.[4] "This script goes through Sound and TextGrid files in a directory, opens each pair of Sound and TextGrid, calculates the pitch maximum of each labeled interval, and saves results to a text file" [7].

The implementation of this script caused another problem and some modifications to the script were made.

The inputs to this script are:

1. WAV files,
2. TextGrid annotation files.

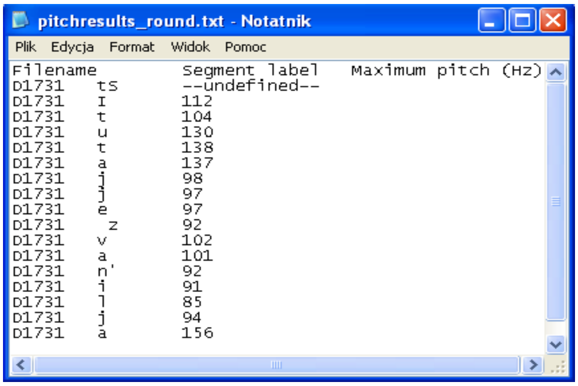
The problem was that the available annotation files were in BLF format. That fact prompted a design of a new conversion algorithm. The approach taken was to convert the PHO files with monotone into TextGrid files. That choice was made, in contrast to converting BLF to TextGrid, because the script converting BLF phoneme set into the Polish Female Voice database already existed.

[4]I am grateful to the author of the Praat script, Mietta Lennes, for this freeware application, which is distributed under the GNU General Public License.

The Praat pitch extraction file produces one TXT file with the pitch values of all the phonemes in the files in the directory. The output “pitchresults.txt” file contains following information:

1. filenames of the files in the directory,
2. labels,
3. maximum pitch values of the labeled intervals in Hz.

The *pitchresults* file for one file in a directory is shown in Figure 5. Because of the *pitchresults* format, the automatic integration of the original pitch values was left untouched at the beginning, creating a new transition phase in developing the full ACCS synthesis system called *Semi-ACCS synthesis* in which data about pitch values were put into the PHO files manually (cf. Figure 2). This step was later automatised.



```
Filename          Segment label    Maximum pitch (Hz)
D1731    tS       --undefined--
D1731    I        112
D1731    t        104
D1731    u        130
D1731    t        138
D1731    a        137
D1731    j        98
D1731    j        97
D1731    e        97
D1731     z       92
D1731    v        102
D1731    a        101
D1731    n'       92
D1731    i        91
D1731    l        85
D1731    j        94
D1731    a        156
```

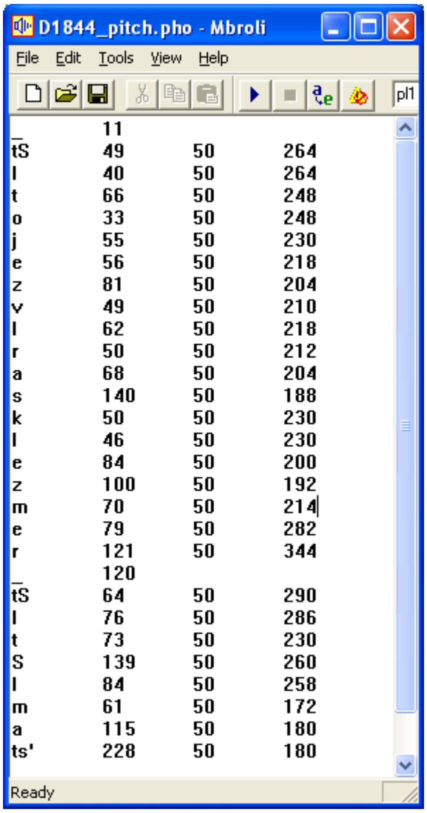
Figure 5: Pitchresults file generated by max_pitch Praat script.

6.4.5. Inclusion of pitch values into MBROLA PHO file

Although at the beginning the extracted pitch values with the use of *max_pitch* Praat script were put into the MBROLA PHO files manually in the procedure called Semi-ACCS synthesis, finally, an automatic inclusion of pitch values into MBROLA PHO files was developed. This procedure takes the *pitchresults* file generated by the modified *max_pitch* Praat script as an input. As described above the *pitchresults* file contains the names of all the WAV/TextGrid files (the WAV and TextGrid file names are identical) in a directory, labels and the maximum pitch for segments of these files. The problem was solved by dividing the *pitchresults* file into separate PITCH files in which there are only filenames, labels and pitch values for each file in a directory. Therefore, the PITCH files got the same length as TextGrid files. Similarly, PITCH files and MBROLA PHO files were almost identical. The difference was in the automatically generated first and last pauses in the PHO files. These pauses were removed by the Perl script and the PITCH and MBROLA PHO files were made the same length. Finally, the inclusion script takes:

1. from MRBOLA PHO files with monotone:
    1. phonemes,
    2. duration of these phonemes,
    3. pitch position equal 50.

2.from PITCH files: pitch values.



```
_     11
tS    49    50    264
I     40    50    264
t     66    50    248
o     33    50    248
j     55    50    230
e     56    50    218
z     81    50    204
v     49    50    210
I     62    50    218
r     50    50    212
a     68    50    204
s     140   50    188
k     50    50    230
I     46    50    230
e     84    50    200
z     100   50    192
m     70    50    214
e     79    50    282
r     121   50    344
_     120
tS    64    50    290
I     76    50    286
t     73    50    230
S     139   50    260
I     84    50    258
m     61    50    172
a     115   50    180
ts'   228   50    180
```

Figure 6: The automatically generated PHO file in ACCS synthesis.

Before printing new PHO files, a simple pitch emulation for the MBROLA female voice was introduced: the male pitch values are multiplied by two, because the original recordings are for a male voice and the diphone database is for a female voice. Furthermore, for phonemes where pitch value is undefined (the phonemes are voiceless), the script prints only phonemes and duration. The pitch pair (pitch position and pitch value) are left out. The automatically generated PHO file by the ACCS synthesis script is shown in Figure 6.

Figure 7 shows a waveform and a pitch contour of a human utterance *Michał podśmiewał się z kolegi, który* *dostał jedynkę ze sprawdzianu* (Michał laughed at a pupil who failed a test.) derived from the corpus. Below there is a waveform and a pitch contour of the same utterance, but synthesised with the ACCS synthesis procedure. Comparing both pitch contours there is not a big difference between them, which indicates that the prosody of the synthesised speech can be very human-like.

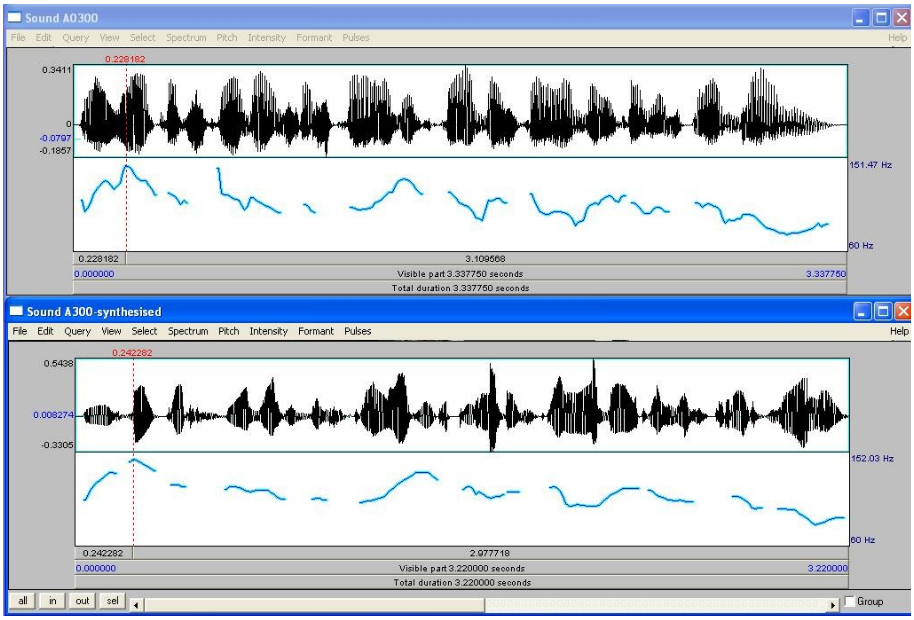

Sound A0300
File Edit Query View Select Spectrum Pitch Intensity Formant Pulses Help
0.228182
0.3411
0
-0.0797
-0.1857
151.47 Hz
60 Hz
0.228182
3.109568
0.000000
Visible part 3.337750 seconds
3.337750
Total duration 3.337750 seconds
Sound A300-synthesised
File Edit Query View Select Spectrum Pitch Intensity Formant Pulses Help
0.242282
0.5438
0.008274
-0.3305
152.03 Hz
60 Hz
0.242282
2.977718
0.000000
Visible part 3.220000 seconds
3.220000
Total duration 3.220000 seconds
all in out sel
Group

Figure 7: A waveform and a pitch contour of a human utterance and its synthesised equivalent using the ACCS synthesis.

6.4.6. BLF to TextGrid transformation software

BLF to TextGrid transformation software was developed because *max_pitch* script requires annotation files in TextGrid format. Algorithms converting the set of phonemes used in annotation files into the set of phonemes used by the Polish Female Voice had already been developed for creating PHO files. Therefore, in the ACCS speech synthesis system TextGrid files are made on the basis of PHO files, not the original BLF files.

7. Conclusion and future strategies

In this study, the development of a speech synthesis component for use in speech perception tests for cochlear implants in children was described and a prototype implementation was developed. An overview of available resources was provided. Preliminary diagnostic evaluation of the system was carried out with successful results. Also informal tests of comprehension of speech were done with very good results. The next step is to carry out speech output evaluation tests according to criteria and methods outlined in [8] and [9].

BIBLIOGRAPHY

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

[2] Bachan, J. & Gibbon, D. 2006. Close Copy Speech Synthesis for Speech Perception Testing. In: *Investigationes Linguisticae*, vol. 13, pp. 9-24. <http://www.staff.amu.edu.pl/~inveling/pdf/Jolanta_Bachan_Dafydd_Gibbon_INVE13.pdf>

[3] Demenko, G., Grocholewski, S., Wagner, A. & Szymanski M. 2006. Prosody annotation for corpus based speech synthesis. In: *Proceedings of the Eleventh Australasian International Conference on Speech Science and Technology*, pp. 460-465. Auckland, New Zealand.

[4] Gibbon, D., Bachan, J. & Demenko, G. Forthcoming. Syllable timing in Polish: results from annotation mining.

[5] Szklanny, K. & Masarek, K. 2002. PL1 - A Polish female voice for the MBROLA synthesizer. Copying the MBROLA Bin and Databases. <http://tcts.fpms.ac.be/synthesis/mbrola/mbrcopybin.html>, accessed 2006-11-25.

[6] Dutoit, T. 2005. The MBROLA project. <http://www.tcts.fpms.ac.be/synthesis/mbrola.html>, accessed 2006-11-30.

[7] Lennes, M. 2003. Praat script – collect_pitch_data_from_files.praat. <http://www.helsinki.fi/~lennes/praat-scripts/public/collect_pitch_data_from_files.praat>, accessed 2006-02-18.

[8] Gibbon, D. & Moore, R. & Winski, R. 1997. *Handbook of Standards and Resources for Spoken Language Systems*. Berlin: Mouton de Gruyter.

[9] Gibbon, D. & Mertins, I. & Moore, R. 2000. *Handbook of Multimodal and Spoken Dialogue Systems: Terminology, Resources and Product Evaluation*. New York: Kluwer Academic Publishers.