DUTCH TEXT-TO-SPEECH AIDS FOR THE VOCALLY HANDICAPPED

B. Van Coile and J.P. Martens

Laboratory of Electronics and Metrology, State University of Ghent, Belgium
* Research Associate of the National Fund for Scientific Research

Abstract

This paper describes two Dutch talking communication aids. They offer full text-to-speech capabilities and produce a highly intelligible and naturally sounding output. We focus on the speech synthesis strategy, and its implementation in hard- and software. Much attention is paid to special development tools which have enabled us to perform the software development independently of the hardware.

1. INTRODUCTION

The speech aids described in this paper must be considered as a spin-off of our fundamental research in the domain of text-to-speech synthesis. We have developed a text-to-speech system consisting of three parts: a linguistic, a phonetic and a synthesizer part:

1. The linguistic part creates the phonetic transcription of the input text. It also adds lexical, syntactic and semantic information to the phonetic representation.

2. The phonetic part uses the information made available by the linguistic part to produce speech parameters. It incorporates strategies for the creation of a correct prosodic pattern and for the synthesis of correct spectral characteristics.

3. The synthesizer part uses the speech parameters to drive a speech synthesizer.

2. THE LINGUISTIC SECTION

To a large extent, the linguistic section of our present system deals with the conversion of the input text into its phonetic transcription. This orthographic-to-phonetic conversion is mainly accomplished by means of pronunciation rules. For example, the following rule is applicable to Dutch: "At the end of a word, a character [d] is pronounced as /t/." A collection of such context dependent rules is called a (linguistic) grammar.

In order to facilitate and to speed up the development and the testing of such grammars, we conceived a flexible rule development tool named DEPES [1]. Some of its features will be discussed in section 6. The development tool was used successfully for the development of the rule component of our text-to-speech system. This rule component consists of several grammars which mainly perform the following tasks:

- deal with special character strings such as digit strings, data, time indications, abbreviations, etc.
- make a grapheme-to-phoneme conversion
- perform a syllabification of the words
- assign lexical stress.

The system also includes some grammars with heuristics:

- to determine the major syntactic boundaries
- to introduce pauses correctly
- to determine the important words of a message

The present system uses nearly 500 rules. About 300 of these rules deal with the orthographic-to-phonetic transcription of single words. The necessary linguistic knowledge was extracted from a phonetic dictionary of the 10,000 most frequent words of Dutch.

The rule component also refers to three dictionaries which contain:

- high frequency words together with part-of-speech information
- Dutch abbreviations
- exceptions to the rule-based strategy (less than 1200 entries)

The complete system is able to perform the orthographic-phonetic transcription with a high accuracy. It has been estimated that less than 5% of the words in an unrestricted text show some pronunciation defect.

3. THE PHONETIC SECTION

The phonetic section performs the segmental and prosodic synthesis. It includes modules:

- to create a correct durational pattern
- to synthesize an adequate intonation contour
- to create good spectral parameters.

The system uses a durational model for Dutch that attributes a well estimated duration to each phoneme. In
order to discover durational rules for continuous speech we
analysed a Dutch text with a total length of more than 8
minutes [2]. This text was read by the same female speaker
which was used for the development of the segmental syn-
thesis part (see further). Rules were developed to explain
the bulk of the durational variations observed in the text.
They account for phenomena such as short/long opposition
of Dutch vowels, word final lengthening, prepausal len-
thening, the influence of prominence, etc. The present version
of the model uses a limited set of durational rules that ac-
counts for 81% of the total variance observed in phoneme
durations.

The phonetic section also creates a correct intonation
contour for the message to be synthesized. The system uses
a strategy based on a study by 't Hart and Cohen (1973) [3].
The intonation contours are described in terms of standard-
ized pitch movements. A limited number of rules specify
how these movements can be combined to create an intona-
tion contour for the whole message. The rules take into ac-
count the number and the location of the dominant words
and the major syntactic boundaries. Our system uses a
DEPES grammar to create the linguistic description of an
adequate intonation contour.

In our approach to speech synthesis, we use an invento-ory of diphones supplemented by a minority of triphones
to compose any message. All segments were taken from
isolated words spoken by a female native speaker of Dutch.
An LPC technique was used to analyse the words. Then,
the segments were extracted semi-automatically [4] and
stored in the segment inventory. During the text-to-speech
conversion, the inventory of segments is used to perform
the segmental synthesis. This involves the following steps:

- subdividing the phonetic transcription into segments.
- consulting the segment inventory to create a sequence
  of speech parameters.
- stretching and shrinking the interval between suc-
  cessive parameter sets according to the durational
  model.
- creating a set of (time,pitch) breakpoints describing
  the fundamental frequency contour.
- transmitting the parameters to the synthesizer part.

4. THE SYNTHESIZER SECTION

The speech parameters produced by the phonetic sec-
tion are supplied to an artificial speech production model.
The model we adopted is the familiar source-filter model
known from LPC vocoders [5]. Per frame, the synthesizer
must receive the following information:

- 12 reflection coefficients characterizing the all-pole fil-
ter
- the Voiced/Unvoiced (V/UV) parameter determining
  the excitation source
- the required speech amplitude (rms value)
- the frame length in samples

The pitch of voiced frames is computed from the set of
(time,pitch) breakpoints produced by the phonetic section.
The excitation amplitude is computed from the required
speech amplitude and the reflection coefficients according
to a procedure described in [6]. Due to the durational
model, the frame length can vary between 40 and 120 sam-

The inventory of speech segments nearly counts 1500
segments. Therefore, it is important to employ an efficient
coding scheme for storing the parametric representations.
We have adopted an optimized 48 bit codeward representa-
tion of the reflection coefficients based on the recommenda-
tions of [7]. Furthermore, a city block distance criterion in
the space of codewords was employed to decide whether or
not to repeat the previously encoded reflection coefficients.
Excellent results were obtained with a baudrate of 3000
bit per second of speech. The entire inventory of speech
segments could be stored in less than 64 KByte of memory.

5. HARDWARE IMPLEMENTATION

If one wants to meet the individual demands of dis-
abled customers, it is important to support different kinds
of input devices. Consequently, it was decided to separate
the input device from the speech generator module, and to
connect the two via a standard serial (RS232) or parallel
link.

Moreover, since potential users of speech aids not al-
ways have the mental and motoric skills to compose words
and sentences out of single characters, one must be able to
support different kinds of input strategies such as text-to-
speech, symbol-to-speech and others. This can be accom-
plished with programmable hardware only.

Obviously, text-to-speech synthesis involves different
kinds of data processing. The linguistic and phonetic sec-
tion are mainly performing logical inferences and database
retrieval operations, the synthesizer section is merely exe-
cuting multiplications and additions. Given the different
nature of these operations we have designed a CPU-board
incorporating a general purpose microprocessor (the Intel-
8086) and a special purpose signal processor (the TMS320-
10 from Texas Instruments). A general outline of our sys-
tem is depicted in fig. 1.

While the Intel-8086 and the TMS320 are running, they
can exchange information over the inreg and outreg data
ports. The linguistic and phonetic processing is performed
by the Intel-8086, the synthesizer program is running on
the TMS320. However, before the synthesizer can start,
the synthesizer program must be downloaded from the on-
board 16 KByte EPROM into the TMS program mem-
ory. This can be accomplished by resetting the TMS320,
opening the data transceivers next to outreg, and selecting
the Intel-8086 address bus by switching the multiplexer to
the right position. The on-board EPROM also contains

dictionaries: one with abbreviations and another with

EUROSPeECH '89, Paris, France, September 1989
high frequency words. The elementary speech segments and the pronunciation knowledge are stored in the two 512 kbit EPROMS on the memory/DAC board. The remaining memory is for data variables (RAM) and Intel-8086 executable code (EPROM).

The two printed circuit boards, a high quality loudspeaker, and two 6 Volt rechargeable batteries are assembled in an esthetic plastic cabinet. The 25-pin connector on the back of the cabinet is wired to the serial (RS232) or the parallel data port on the CPU board. The battery terminals and the ON/OFF and RESET lines are also wired to that connector. In this way, an external source can use the batteries as a power supply, and the ON/OFF and RESET lines to switch on and off the speech module. The internal loudspeaker can produce a maximum power of about 4 Watt rms, but it is switched off from the moment an external loudspeaker or headphone is connected.

![Figure 1: General outline of the speech module](image)

6. SOFTWARE IMPLEMENTATION

One of our major goals has been to develop most of the software using high-level language development tools. In this section we discuss in more detail two of these tools, namely DEPES [1] and DSPL [8]. They were both developed in our laboratory.

When text-to-speech rules are hard-coded into a computer program, it is difficult to make changes. In order to overcome this problem, we conceived a flexible rule development tool named DEPES (Development Environment for Pronunciation Expert Systems). It offers a powerful knowledge representation language which was carefully designed in order to combine flexibility with ease of use. A full description of the DEPES language is beyond the scope of this paper. Still, some of its features are listed:

- The DEPES language is based on the well known formalism of generative phonology [9].
- The structure of a DEPES module very much resembles that of a Pascal module.
- Rules operate on a user-definable multi-layered data structure. They can match patterns across the borders of different layers, and modify information in these layers.

- The user has the ability to select the proper inference strategy (do until .., from right to left, ..) at the proper time.

The DEPES tool incorporates a compiler, a linker, a debugger and a dictionary tool. It was used to create the necessary Pascal modules for the rule based components of the linguistic and phonetic sections. By means of DEPES, new and enhanced grammars can easily be incorporated into our system.

Since there was no high-level language support available for the TMS320-10, we have developed a Pascal-like language called DSPL (Digital Signal Processing Language) [8]. It was developed in cooperation with an industrial partner who is now distributing a compiler, a linker and a library on PC and VAX [10]. Some of the most important features of DSPL are listed below:

- DSPL very much resembles standard Pascal. However, there are no recursive procedures, and no strings, sets or dynamic memory.
- Apart from integers and reals, DSPL fully supports operations with 16-bit fixed point fractional numbers (between $-1$ and $1 - 2^{-15}$).
- In DSPL, the user has to distinguish between internal RAM variables (declared in VAR sections) and external program memory variables (declared in PVAR sections).
- Pointers in DSPL are implemented as 16-bit addresses. They can be manipulated like integers, and used for efficient indirect addressing operations.
- The logical operators OR, AND, EXOR and NOT can also act on integers. They are performed bitwise then. Trivial application of this are bit testing and masking.
- There are special instructions for installing and inhibiting interrupt procedures. One can prevent procedures from being interrupted by an interrupt procedure. The interrupt is then acknowledged after the procedure is executed.
- Processor-dependent instructions like IN, OUT, .. are used to perform I/O operations, and to make fully available the capabilities of the TMS320.

The DSPL compiler supports separate compilation units which can import and export variables and procedures, and the linker computes a suitable overlay of variables in internal RAM (only 144 words available in TMS320-10). The library supports standard real arithmetic operations, and many standard functions which are usually available in Pascal languages.

The 12-th order synthesizer of our text-to-speech system has been entirely implemented in DSPL. It requires no more than 700 words of program memory and 50 words of internal RAM. The output sampling frequency is 10 kHz.

---

1 TELEVIC nv, Leo Bekaertlaan 1, B-8701 Izegem (Belgium)
To conclude we recall that the linguistic and phonetic section are running on the Intel-8086. They were almost entirely developed using the powerful DEPES language. Some routines were written in Pascal-86 [11]. Only a few basic I/O operations, some hardware initializations and particular memory allocations were programmed in Assembly-86 [11]. The speech synthesizer is written in DSPL. Therefore, our software development can be considered independently of the hardware.

7. TWO PARTICULAR REALIZATIONS

In this section we briefly discuss two speech aids which were developed using our speech module.

Our first realization is BLISS-VOICE, a speech aid for symbol-to-speech communication. It consists of a wheelchair table comprising a large number of symbolic and/or pictographic representations of ideas, and a speech module interpreting what is appointed on the table. The BLISS-VOICE software is intelligent enough to produce a naturally sounding sentence given a sequence of ideas. It contains a finite state automaton which is driven by the incoming requests. For instance, if a noun is to be pronounced, the plural state variable will determine whether the singular or the plural form is to be generated. In case of a verb, the proper conjugation and tense are derived from other state variables. After the speech output is generated, a state transition may take place. The sequence /I/, /'past' function/, /to see/, /5/, /horse/, would be correctly pronounced as /I have seen 5 horses/.

Basically, BLISS-VOICE uses a fixed vocabulary, but thanks to the text-to-speech synthesis strategy, one can always overcome the problems of vocabulary limitations. If single character representations are available on the table, they can be used to form a new word. It then only takes function keys to form and to recall this word. It is even possible to store newly formed words into spare positions on the table, and to recall them later.

A second realization is PC-VOICE, a system offering full text-to-speech capabilities in Dutch. It has to be connected to a host system (e.g. a personal computer) via a serial interface. It can read aloud any ordinary spelled message in Dutch. Moreover, it correctly pronounces digit strings, telephone numbers, date and time indications, abbreviations and alphanumeric strings. In order to solve pronunciation ambiguities, ordinary spelled sentences may be supplemented by additional information such as stress markers.

An additional feature of PC-VOICE is that it supports the use of personal dictionaries. The user can create his own dictionary on a personal computer and add it to the pronunciation knowledge already available. Such a personal dictionary may include both ordinary spelled and phonetically written translations. Consequently, it can be used to expand uncommon abbreviations, to define the pronunciation of foreign words, to correct mispronunciations of words, etc.

In order to be useful as a reading machine for visually handicapped persons, PC-VOICE is delivered with the necessary software support for an IBM-PC or compatible. Thanks to this software, the text on the PC screen can be read even during sessions with standard software such as text processing or data base programs.

8. CONCLUSION

We have discussed the principles underlying the artificial generation of speech, and described two speech aids capable of producing naturally sounding speech in Dutch: BLISS-VOICE for symbol-to-speech, and PC-VOICE for text-to-speech communication. Both aids are commercially available.

9. ACKNOWLEDGEMENTS

This work was supported by the Belgian Ministry of Public Health, and by the Van Goethem-Brichant Foundation (Belgium).

10. REFERENCES


2Technologie et Revalidatie vzw, Postbus 128, B-9000 Gent (Belgium)