Annales UMCS Informatica AI 2 (2004) 315-323

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Computer speech echo-corrector

Mariusz Dzieńkowski[a*], Wiesława Kuniszyk-Jóźkowiak[b],
Elżbieta Smołka[b], Waldemar Suszyński[b]

[a]*Management Department, Lublin University of Technology,*
Nadbystrzycka 38, 20–618 Lublin, Poland
[b]*Institute of Physics, University of Maria Curie-Skłodowska,*
pl. Marii Curie-Skłodowskiej 1, 20-031 Lublin, Poland

Abstract

The "Echo" method, elaborated by Adamczyk, which is commonly applied in many logopaedic clinics, requires the use of the delayed auditory feedback. The devices which perform this task may nowadays be replaced with a typical computer equipped with a sound set and delaying programme. In the article the authors present a real-time application, implemented on the basis of standard sound functions supplied by Microsoft in Windows systems.

1. Introduction

Delayed auditory feedback has been applied in the therapy of stuttering people for a long time. Numerous studies have shown the positive influence of that factor on speaking process. Stuttering people who speak into a microphone and hear delayed sounds of their own speech in headphones or a speaker begin to speak fluently at a slower pace. It was stated that the optimum delay time for therapy should amount to no less than 0.1 sec and no more than 0.2 sec.

Initially the devices generating echo worked on the basis of tape-recording the speech signals and reproducing them after the time of 0.2 sec. At present the delay is produced with the application of digital shifting registers [1]. Sound echo may also be obtained directly with the use of a computer which is equipped with a sound set. It is an optionally applied part of a computer diagnostic-therapeutic system elaborated by the authors (Fig. 1) [2]. What is also planned is the incorporation of visual feedback into the programme [3].

* Corresponding author: *e-mail address*: m.dzienkowski@pollub.pl

Fig. 1. Computer programme for diagnosis and therapy of speech disorders with speech echo-corrector active

2. Recording and playback of audio data in real time

Recording and playback of sound in real time may be performed with the use of regular personal computers which are equipped with a microphone, sound card, speakers or headphones. The condition of performance is to obtain possibly the least inexpedient delay introduced by the equipment and system software. The sound card should work in the "full duplex" mode, which allows for simultaneous registration and playback of audio data and ensures quick communication between the input circuits, memory and output circuits (Fig. 2). These tasks are performed by sound devices equipped with a controller which is linked with two separate converters: analogue-digital (recording) and digital-analogue (playback). The controller sends buffers with data directly into and from primary memory by means of the DMA channel (Direct Memory Access). The role of the software is to send instructions to the controller and memory allocation to audio data placed in the buffers.

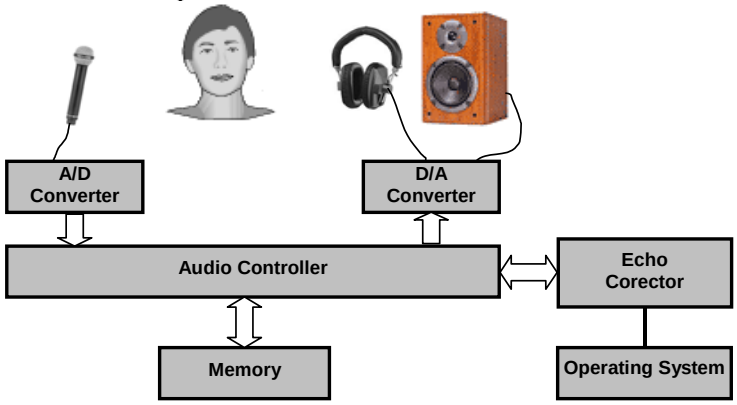

Fig. 2. The principle of speech echo-corrector operation

The audio input and output management in real time may be realised by means of:

- low-level WinApi audio functions, made available by Microsoft in their operating systems beginning with Windows 3.1,
- DirectSound mechanisms available in the DirectX package [3].

During the implementation of the echo-corrector standard WinApi functions were used, which can be found in the WinMM.dll library of Windows. The method, in spite of a rather significant equipment delay, does not require installation of extra libraries. The minimum requirements of the programme are: Pentium 100, 32 Mb RAM, dual channel sound card and Windows 95.

3. Operation algorithm of the speech echo-corrector

The presented echo-corrector was implemented with the aid of Delphi 5.0 visual tool made by Borland. The main delaying algorithm was located in a separate module as a so-called Thread Object in order to allow for simultaneous recording and playback operations to be realised in separate threads. Due to this the programme does not appropriate the whole system.

Initially, in the main module of the programme, wave format is defined for the operations of speech signal recording and playback (Fig. 3). The description of the wave format is contained in the structure of WAVEFORMATEX:

- **wFormatTag** – defines the type of audio format (Wave_Fromat_PCM),
- **nChannels** – the number of audio channels (1 = mono for speech),
- **nSamplePerSec** – the sound sampling frequency (22 050Hz for speech),
- **wBitsPerSample** – the size of each sample in bits (16 bits for speech),
- **nAvgBytesPerSec** – the average number of bytes which are processed by the programme in real time (nSamplesPerSec * nBlockAlign = 44 100Hz),
- **nBlockAlign** – the number of bytes per each sample (nChannels * wBitsPerSample / 8 = 44 100 Hz).

Subsequently, the programme checks whether the system can start the activities of recording and playback. Two functions serve that purpose: input and output, functions usually applied for opening the device and checking the sound format. The functions are the following: **waveInOpen** (LPHWAVEIN *handle,* UINT *idDev,* LPWAVEFORMATEX *format,* DWORD *callback,* DWORD *callbackInstance,* DWORD *open*) and **waveOutOpen** (LPHWAVEIN *handle,* UINT *idDev,* LPWAVEFORMATEX *format,* DWORD *callback,* DWORD *callbackInstance,* DWORD *Open*), where:

- *handle* – is the pointer to buffer, which stores the handle for identification of the opened input or output device; when the value of WAVE_FORMAT_QUERY is specified, the parameter may have 0 value,
- *idDev* – is the identifier of the opened sound devices,
- *format* – is the pointer to WAVEFORMATEX structure, which contains the description of the recorded or played sound format,

- *callback* – is the pointer to: the fixed callback function, event handle, window handle, or thread identifier, which serve the processing of the messages related to the recording or playback processes,
- *callbackInstance* – is the user value relayed to the callback mechanism (usually 0),
- *Open* – is the opened device flag (CALLBACK_EVENT, CALLBACK_FUNCTION, WAVE_FORMAT_QUERY, etc.).

The above-mentioned functions are called at the beginning of the algorithm in order to check whether the device accepts the defined wave format. An example handling function, which does not, however, open the input device, should be activated in the following way: **waveInOpen** (0, idDev, format, 0,0,WAVE_FORMAT_QUERY). The last parameter is the flag which commands the system to check if the device supports the given format. If it does not, an error message is generated and *false* value is returned. Otherwise the device may be ready to record. Calling the function with the following parameters set: **waveInOpen** (*@handle, idDev,* WaveFormat *@format,* (DWORD) HANDLE*(event),* 0, CALLBACK_EVENT) opens the device. (DWORD) HANDLE*(event)* is the pointer to the callback function, which uses the event handle in the described system. The function is **WaitForSingleObject().** After the audio devices have been successfully opened, the programme initiates buffers – separately for audio input and output. They are located in WAVEHDR structures, where the area values are established: *lpData* and *dwBufferLength.* Then buffers are prepared for application by means of **waveInPrepareHeader()** and **waveOutPrepareHeader()** functions. The ready buffers, associated with the waveform-audio input device, are sent there by means of **waveInAddBuffer()** function.

During the recording the audio data are located in 4 buffers, marked respectively with numbers from 0 to 3. The playback process is also handled by a 4-buffer queue.

The **waveInStart**() function begins the sound recording operation. It is followed by the main part of the programme, which is constituted by a *while-do* loop, launched and terminated by the user with the aid of the S*tart/Stop* button. Inside the loop 3 operations are performed:

- downloading of audio data into the input buffer,
- copying of the input buffer content into the output buffer,
- data output from the output buffer into the output of the audio device.

The **WaitForsingleObject()** function, which is located in the loop, inhibits the operation of the application until the event object changes from the nonsignal to the signal state. The event is in the sleep state during the process of sound recording into the buffer. When the buffer is completely filled with samples, the state changes from nonsignal to signal. Then the **ResetEvent()** procedure releases all the threads and the application operation continues. The buffer with

the saved data is goes to the application, where its whole content is copied into the output buffer (**CopyMemory()**). Then, by means of the **waveOutWrite()** function, the output buffer is sent to the playback device.

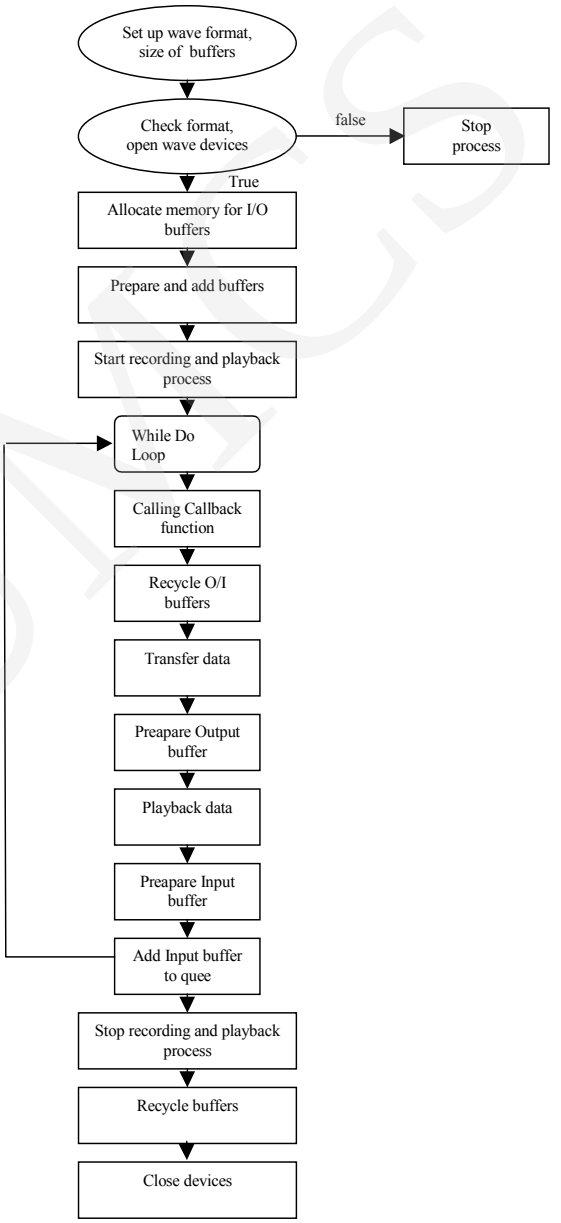

Fig. 3. The flow chart of the delaying algorithm of the speech echo-corrector

The exact description of the loop operation is contained in the few following points:

- call **WaitForSingleObject()** function, signalling the performance of the job on the buffer,
- start the **ResetEvent()** procedure, which manually sets the event state as unsignal,
- recycle the input buffers with the **waveInUnprepareHeader()** function and of the output buffers with the **waveOutUnprepareHeader()** function
- apply the **CopyMemory()** function to copy the content of a full buffer into an output buffer
- prepare output buffer for use with the aid of the **waveOutPrepareHeader()** function
- send ready output data blocks to the playback device (the **waveOutWrite()** function)
- prepare input buffer with the use of **waveInPrepareHeader()**
- add input buffer to the recording device using the **waveInAddBuffer().**

After the *while-do* loop has finished its operation, the **waveInStop()** function is performed, stopping the recording and playback process. The whole system of buffers is released by calling **waveInReset()** and **waveOutReset().** In the last stage of the programme operation, the headers are recovered (the **waveInUnPrepareHeader()** and **waveOutUnPrepareHeader()** functions) and the memory allocated to them is released. Finally, the **waveInClose()** and **waveOutClose()** functions are called, which close the audio devices.

Audio data are located in the two buffer systems: input and output. The size of each data block is related to the delay time set by the user. For 100 msec it amounts to 4410 bytes. From this it follows that during 1 sec, 22050 two-bytes samples are registered, which results in 44 100 bytes per sec. In the research the optimum values have been assumed for recording and playback of speech signals (sampling frequency of 22050 Hz, sound resolution of 16 bites, number of channels – 1). All buffers have to be the same size. The bigger the buffer, the greater delay time may be obtained. Setting too small a size of audio data block and too small a number of buffers may prove dangerous to the recording and playback processes, the reason being the device latency, which causes interruptions in signal playback. The latency depends on the CPU and operating system speed and varies for particular computers. In the case when the buffer size is too small, the playback time is shorter than the latency, which results in the sound being non-fluent. Thus, the total pause between the recording and playback is the sum of the times of filling the input buffer and device delay.

4. Possibilities and development of the programme for speech disorder correction

The presented programme makes it possible to choose the delay time, fluently adjust the sound level and record exercises with the echo on the hard disk as wave-files. The delay adjustment is done in strokes, in 25 msec intervals within the range of 75 to 200 msec. At present the possibility of incorporating visual feedback into the programme is being worked in (Fig. 4).

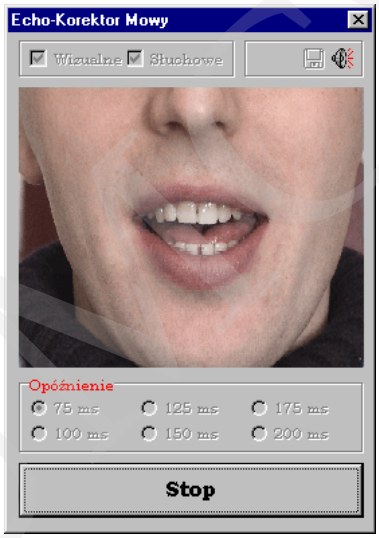

Fig. 4. Speech echo-corrector with visual feedback

In the first visual version of the application, apart from the delayed speaker's own speech signal to be heard in headphones, the patient would also be able to observe an animated face with its lips moving on the computer screen. In this stage the width of lip opening depends on the sound level calculated in the particular moment. The programme users are also free to choose the correction method themselves. They have a choice of auditory, visual and auditory-visual correction [2]. Work on the programme was done in a few stages. Initially, a film recording of a speaking person's face was made with a digital camera. Then an appropriate fragment of the film was chosen and with the use of a specially-designed programme it was cut into separate frames, which were recorded on a computer disk as graphic files. In the next stage frame sequences were isolated form the recording and stored in appropriate catalogues. In the end, the auditory programme for speech disorder correction was modified. In the programme, the pictures of the face with varying lip opening, which had been prepared earlier, were assorted to appropriate calculated sound levels.

5. Summary

The computer programme for speech disorder correction elaborated by the authors is used in its auditory version in work with stuttering people. It has been made in two versions: as an independent programme and as an element of a bigger system serving the diagnosis and therapy of speech disorders. Thanks to it, people with speaking problems may practise individually at home, using their home computers and echo-corrector. When working with stuttering patients, therapists noticed their great involvement in performing the exercises with the aid of a computer equipped with the programme.

Acknowledgements

The research was supported by Grant KBN No. 4 T11E 035 22 from State Committee for Scientific Research in Poland.

The authors wish to thank Natalia Fedan for translation of the text into English.

References

[1] Kuniszyk-Jóźkowiak W, Adamczyk B., *Auditory-tactile echo-reverberating stutterers' speech corrector, In Optoelectronic and Electronic Sensors II*, Zdzisław Jankiewicz, Editor, Proc. SPIE, 3054 (1997) 240.

[2] Dzieńkowski M., Kuniszyk-Jóźkowiak W., Smołka E., Suszyński W., *Computer programme for speech impediment diagnosis and therapy*, Annales Informatica UMCS, (2003) 21.

[3] Dzieńkowski M., Kuniszyk-Jóźkowiak W., Smołka E., Suszyński W., *Komputerowe słuchowo-wizualne echo dla celów terapii*, Obliczenia naukowe, Polskie Towarzystwo Informatyczne, (2003) 57, in Polish.

[4] Czyżewski A., Kostek B, Skarżyński H., *Technika komputerowa w audiologii, foniatrii i logopedii*, Exit, Warszawa, (2002), in Polish.