Vision echo

Marek Wiśniewski[*], Wiesława Kuniszyk-Jóźkowiak,
Elżbieta Smołka, Waldemar Suszyński

*Maria Curie-Skłodowska University,
pl. M.Curie-Skłodowskiej 1, 20-031 Lublin, Poland*

Abstract

One of the most frequently used methods in the speech disfluence therapy is the one based on the acoustical echo. A patient, who is a subject to therapy, hears his own voice with some delay and, thanks to that, his speech fluency increases. As it turned out a therapeutic effect can also be achieved by feedback of the speech with other human senses, such as touch or sight.

The article presents a computer program that enables achieving the effect of the visional signal delay from camcorder on a monitor screen.

1. Introduction

The method of therapy of people suffering from disturbances of speech fluency, based on the use of the acoustical echo, appeared very effective. However it requires patience from the patient, because participation in therapeutic sessions is indispensable and may lasts many weeks. It often occurs that some time after finishing the therapy, problems with speech fluency return and a treatment is necessary again. So there exists continuous requirement for new and more effective methods of speech disfluence therapy.

During the study of effectiveness of the acoustical echo method, it turned out that also a visual feedback has therapeutic value [1,2]. In that therapy the patient speaks and sees his own picture on a monitor screen with some delay. The delay value should be no less than 0.1 second and no more than 0.2 second, because only these values have a therapeutic application. Until now testing the effectiveness of that therapy method was very difficult. Construction of a therapy set for this method, further called a vision echo method, with appropriate picture quality and possibility of proper adjustment of signal delay was too expensive. It was the popularization of computers with large computational power and appearance of cheap digital camcorders generating high quality

---

[*]Corresponding author: *e-mail address*: marek.wisniewski@umcs.lublin.pl

picture that gave the opportunity to study that method on a great scale. There is a computer program, named "Kamera", its functioning and construction described below. It enables us to use the vision echo method during the therapy.

2. The set applied in the vision echo therapy

The „Kamera" program must be run in the Microsoft Windows environment with DirectX 9.x library installed. The use of the program is intuitive. The sight of main window is presented in Fig. 1. The program automatically detects devices generating a video signal. It has the ability to set a delay range from 100 ms to 1000 ms, with a step of 40 ms. The obtained vision can be displayed either in the window or in the full screen mode, it is also automatically converted in order to achieve a mirror effect.
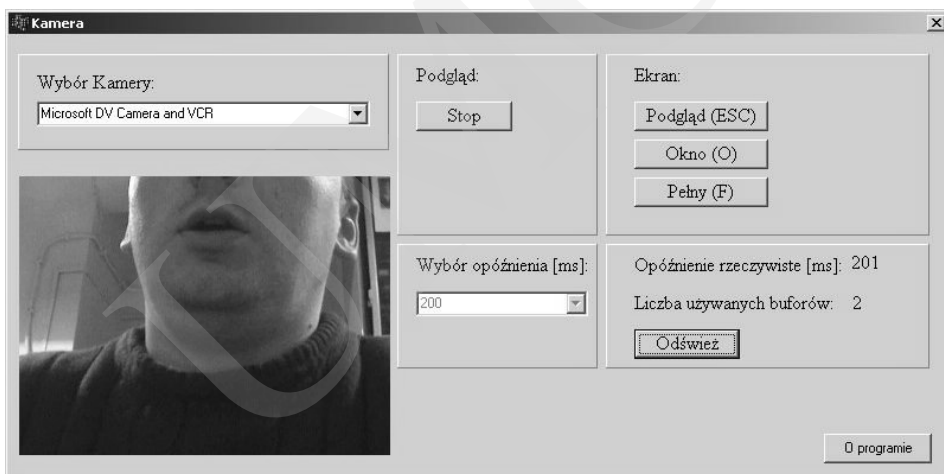

Fig. 1. The main window of the "Kamera" program

During the development of the program the set presented in Fig. 2. was used. The hardware requirements, due to processing a huge amount of data, are quite high. The functioning of the program was tested, with good results, on a PC class computer equipped with a Pentium class processor, with 2GHz clock frequency, 512MB of RAM memory and Windows XP operating system installed. The computer has also a FireWire card installed to which digital camcorder was connected.

Digital camcorder generates vision and sound signals and transmits them in a digital form to the computer. The vision signal has the DV format (Digital Video, IEC 61843 standard) and consists of 50 halfframes per second (interlaced signal), which means that there are 25 frames per second effectively. The frame resolution is 720x576 pixels (PAL standard) and they are encoded using the YUV2 format (16 bits per pixel). Finally, the signal, received by the computer, is

made up of mixed visional and sound data (DV interleaved). The presented program is specially designed to cooperate with such camcorders.
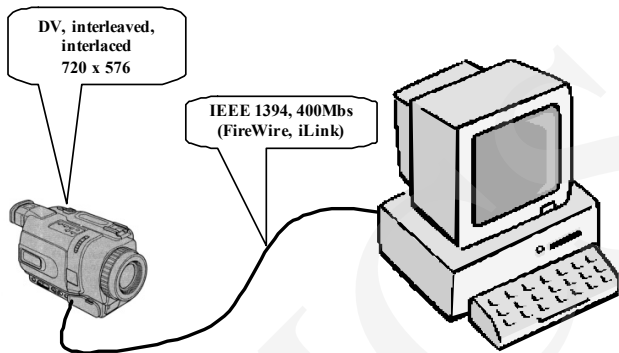

Fig. 2. The set to vision echo method therapy

3. The "Kamera" program structure

The program was written under Borland C++ Builter 6.0 environment. There were elements of DirectShow [3,4] library (part of DirectX library for video processing) utilized.

The heart of the application developed with that library is the concept of a Filter Graph. The filter graph consists of objects called filters, which perform specialized operations on a data stream. Before the moment of the proper application operation, the filters must be properly connected to create the filter graph. The application usually controls a filter graph by calling the methods of helper objects (Filter Graph Managers) and by analysis of the messages generated by the filters in the graph.

The filter graph in the presented application consists of the following objects (Fig. 3.): Capture Filter, DV Decoder Filter, MyTransform Filter and Video Renderer Filter.
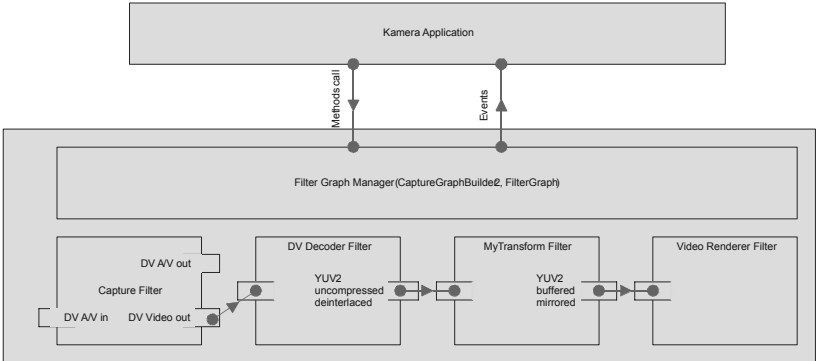

Fig. 3. The "Kamera" application structure

The Capture Filter object represents a video signal source. The object retrieves signals generated by the camcorder and enables it to continue processing. On the output pins of that filter two signals are available: DV Video and DV Audio/Video.

The video data (without sound), in the DV format, are passed down to the DV Decoder Filter. The filter transforms the data from DV format to noncompressed YUV2 form as well as deinterlaces the signal. In the next step, the data are passed to MyTransform filter. This object is the most important part of the "Kamera" program. There, video frames are buffered in order to delay the signal, and they are also transformed to achieve a mirroring effect. Next, the buffered frames are received by the Video Rendered object and are rendered on the screen.

For creation and management of the filter graph, CaptureGraphBuilder2 and FilterGraph objects are used.

The connections between filters are set up using the RenderStream method of the CaptureGraphBuilder2 object.

The FilterGraph object creates a new application thread, where the filter graph runs. It possesses methods to start and stop filter graph. The interface IVideoWindow of that object controls the way the video stream is rendered on the screen (full screen mode, window mode). The IMediaEvent interface is used to intercept messages generated by objects in the filter graph. Currently, messages concerning the camcorder state are handled. The EC_DEVICE_LOST message is used to handle the situation when communication between camcorder and computer is lost.

The connected video devices are automatically detected by the SearchCapDev() function. It uses the CreateClassEnumerator() method from the System Device Enumerator object, to generate a list of video devices.

When the application is run and if the WM_DEVICECHANGE message is intercepted, the SearchCapDev() function is automatically called to refresh the video source list in the main program window.

The DV Decoder, Video Renderer, CaptureGraphBuilder2, System Device Enumerator are the objects included in the DirectX library. The Capture Filter object is dynamically created by the DirectX library during the operation of a given device. The implementation of the author is the MyTransform filter.

4. Functioning of the delay algorithm

After the application is run, it requires selection of the video source device, then the video signal delay must be set and finally the "Start" button pushed (Fig. 1). The "Start" button causes start() function to be called. That function creates the filter graph objects with CoCreateInstance() system function and makes a connection between them. Additionally, there are helpful structures created to keep stream frame time parameters.

The functioning of the delay algorithm is presented in Fig. 4. Immediately after the filter graph is run, the system time is checked with the GetTickCount() function (T1 variable). Next, the first frame is retrieved and moved to the MyTransform filter. There, it is put away to the buffer and the system time is checked once again (T2 variable). The difference between T2 and T1 values indicates the delay between the moment when the first frame is captured by the camcorder and the moment when it reaches the MyTransform filter. This is the first element of the video signal delay. The total delay is the sum of this element and the time of frame processing in the filter graph. The time of the frame processing is calculated on the basis of the T2 variable, which is refreshed every time the frame passes through the filter graph.
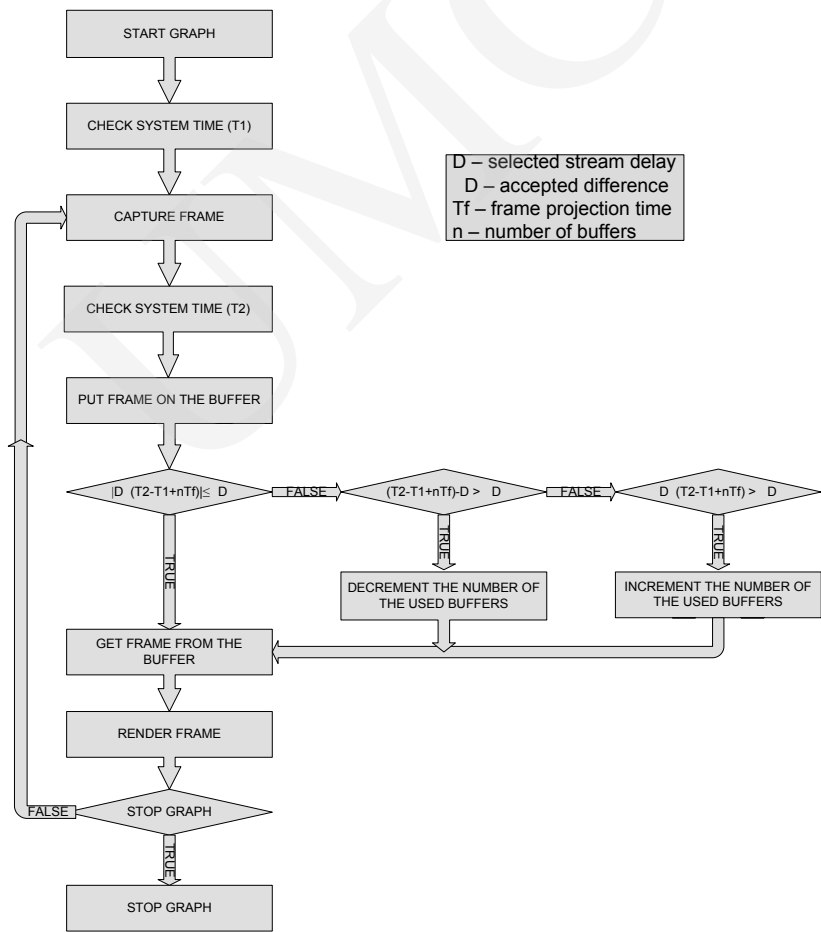

Fig. 4. The video delay algorithm in the "Kamera" program

In the next step, actual delay is compared to the required delay (D variable). The values of these delays can differ from each other by a certain value ΔD, which prevents changing the number of delay buffers too often. When the actual delay of the video signal stays within the ΔD boundaries, the frame is collected from the buffer, passed down to the VideoRenderer object and displayed on the screen. If the difference is too big, then the number of buffered frames is respectively increased or decreased in the next filter graph pass.

Processing of the stream data in the filter graph lasts incessantly until the user stops the filter graph or a camcorder stops functioning.

5. Summary

The "Kamera" program allows for controlling a video signal with the step about 40 ms. From the therapeutic point of view this is sufficient value which permits using the program in practice. However, in the future more accuracy may be required, therefore the application will be further developed to reach this. Additionally, the ability to save a therapeutic session in a form of a mulitmedia files will be added.

In the nearest future the program will be applied during therapy. It allows for testing the influence of the visual echo on speech fluency of the patient.

Acknowledgement

References

[1] Kuniszyk-Jóźkowiak W., Smołka E., Adamczyk B., *Effect of acoustical, visual and tactile echo on speech fluency of stutterers*, Folia Phonia. Logoped., 48 (1996) 193.

[2] Kuniszyk-Jóźkowiak W., Smołka E., Adamczyk B., *Effect of acoustical, visual and tactile reverberating on speech fluency of stutterers*, Folia Phonia. Logoped., 49 (1997) 26.

[3] Microsoft DirectX 9.0 SDK

[4] http://clootie.narod.ru/cbuilder/