Teaching Real-time DSP applications (Voice Removal) with the C6711 DSK and MATLAB

George W.P. York, Christopher M. Rondeau, Dane F. Fuller
U.S. Air Force Academy, CO

Abstract

This paper describes our efforts to teach real-time DSP applications at the undergraduate level. In particular this paper focuses on the voice removal DSP application, removing the lead singer from an audio recording. We find using a real-time DSP application that the students can relate to, like voice removal and other audio special effects, as a course final project highly motivates the students and makes basic DSP concepts more meaningful. While MATLAB simulations are useful for teaching the basic theory, many of these concepts are more easily taught to undergraduates if appropriate real-time demonstrations and laboratory experiences are available. The challenge of transitioning from MATLAB to real-time hardware is often the expense and a steep learning curve for the students. This paper describes a real-time DSP educational platform based around the programming ease of MATLAB and the low-cost Texas Instruments C6711 digital signal processing starter kit. Classroom uses of this platform are discussed.

1. Introduction

While there are many interesting real-time audio DSP applications to choose from, we have found the relatively simple application of voice removal to intrigue the students and offer many engineering tradeoffs, teaching many real-time DSP issues along the way. Voice removal is attempting to remove the lead singer's voice from an audio recording while maintaining the background instruments, useful to make a low-cost karaoke machine. We will first describe the details of a simple voice removal algorithm, then discuss our teaching paradigm, and methods used to easily transition the students from designing in MATLAB, to pseudo-real-time implementation with MATLAB running on the C6711 DSK, to real-time implementation in the C language on the DSK.

2. Voice Removal Algorithm

Voice removal works on the assumption that the lead singer's voice is equally recorded in both the left and right channels (i.e., center stage), while the background instruments and background singers have a phase shift in either the left or right channel (i.e., stage left or stage right)[1]. This assumption usually does not hold for bass frequencies in which all signals tend to be equally recorded in both the left and right channels. Therefore, as shown in Figure 1, the voice removal application must first separate the bass from the rest of the signal. Then the higher frequency signals on the left and right channels are subtracted, resulting in a new signal where the

background instruments and singers are present, but the lead singer is reduced. Finally the bass signal is added back to the instrument signal, resulting in a voice removed signal.

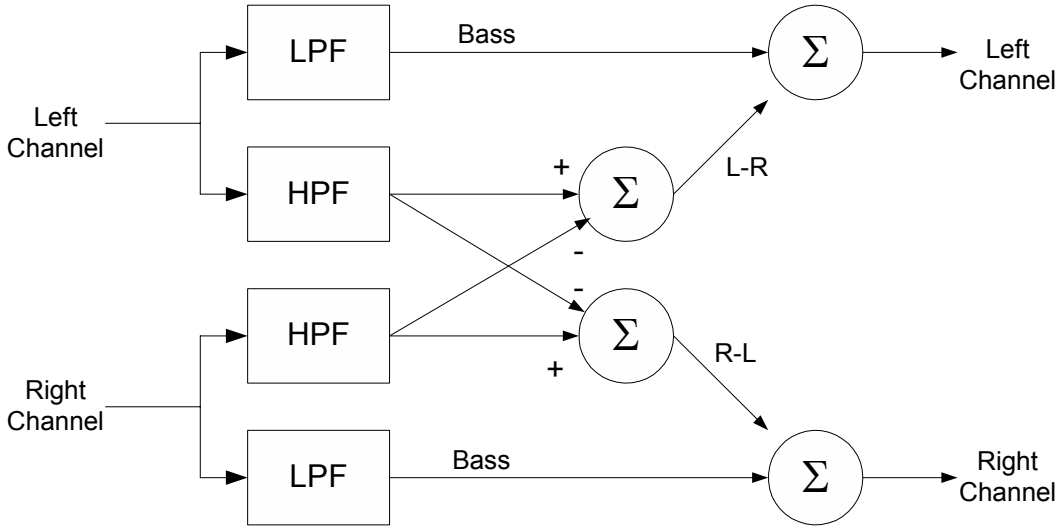

Figure 1. Block Diagram of a Voice Removal System

A side effect of this technique is that it results in a mono signal; stereo is lost. This technique also only works for music recorded following the prior assumptions. This will not work for music recorded with certain audio special effects, like reverb. More advanced voice removal algorithms can be found in the literature, such as in You and Sun.[2]

3. Teaching a Real-time Audio Application: Simulator or Hardware?

Given the desire to teach a real-time voice removal DSP application, how do we impart such concepts to undergraduates? Computer-based simulations can be highly effective in teaching many DSP topics.[3] We can take advantage of the fact that the software package MATLAB[4] and its related toolboxes have become a mainstay in most EE programs. Given our student's familiarity with MATLAB, designing this voice removal system in MATLAB seems to be a natural approach. But while MATLAB can process a recording of a song off-line, our students seem more impressed designing their own system that can voice strip a song in real-time as a CD is played. Ideally then, we need a real-time system to use as a teaching tool.

In the past, proceeding beyond a MATLAB-only simulation to a real-time hardware implementation has been impeded by a very abrupt transition, in terms of both cost and the learning curve of unfamiliar systems and software. By developing a software and hardware bridge between MATLAB and real-time DSP hardware, Morrow et al[5] have made it possible to smoothly and incrementally transition from simulation to a full hardware implementation, all while retaining the versatility and simplicity of the MATLAB display engine. Using this approach, students are able to develop and enhance their own hardware implementation, and easily experiment with several different filter and overall system designs.

4. Hardware Requirements

For the primary DSP hardware, our main criteria were low cost, sufficient processing power, and a versatile software development environment. We chose to construct our educational platform around the Texas Instruments C6711 DSK (DSP Starter Kit), which makes use of the VLIW/SIMD architecture of the TMS320C6711 microprocessor. We have had previous success using the C6711 DSK in the classroom, and believed it could easily perform the voice removal application.[6]

The C6711 DSK has the following advantages[7]:

- An excellent software development environment (Code Composer Studio)
- High processing speed (roughly 1.2 billion instructions per second and 600 MFLOPS)
- Plenty of memory (16 MB)
- Relatively inexpensive ($395)
- Offers both floating point (2 single precision per cycle) and SIMD fixed point (4 16-bit per cycle)

The C6711 DSK's principle disadvantage is that it only has a telephone quality (maximum 8 kHz), single channel codec. To keep up with real-time CD playback, we need to handle at least two 16-bit per sample channels at 44.1 kHz[8]. Fortunately, 16-bit, two-channel simultaneous sampling (f_s = 48 kHz/channel) codec daughter-cards compatible with the C6711 DSK are available. Texas Instruments sells their PCM3003 Audio daughter card for $50[7]. TI is currently replacing the C6711 DSK with the new C6713 DSK for the same price. The C6713 DSK comes with a high-quality 24-bit stereo codec built-in.[7]

The codec is triggered directly by a DSK Timer so that it places no burden on the DSK CPU resources. When the codec captures a pair of samples, the CPU is interrupted, and it reads the two samples from the converter. Likewise, when the CPU is finished processing two output samples, the timer interrupts the codec to output the pair of samples.

5. Educational Requirements

While our students are proficient in programming in MATLAB, they do not all have experience programming in C or C6711 assembly language. Therefore we preferred a tool that not only allowed our students to design and simulate in MATLAB, but also used MATLAB to migrate the algorithm onto the DSP hardware. The desired progression would be as follows.

1. Study the traditional DSP theory
2. Simulate in MATLAB with simulated data (test tones for quantitative analysis)
3. Simulate in MATLAB with a song sampled in real-time by DSK (qualitative analysis)
4. Implement the process in real-time on the TI DSK hardware
5. Repeat to improve the design or to develop new features.

The third step of this process presents a practical problem: How to control the DSK board from the MATLAB simulation environment. We wish to minimize multiple software environments in the interest of time (for students and faculty), therefore we prefer a single development

environment. Fortunately, Morrow et al[5] developed a direct MATLAB to DSK interface and provides this software for free for educational purposes at http://eceserv0.ece.wisc.edu/~morrow/software/.

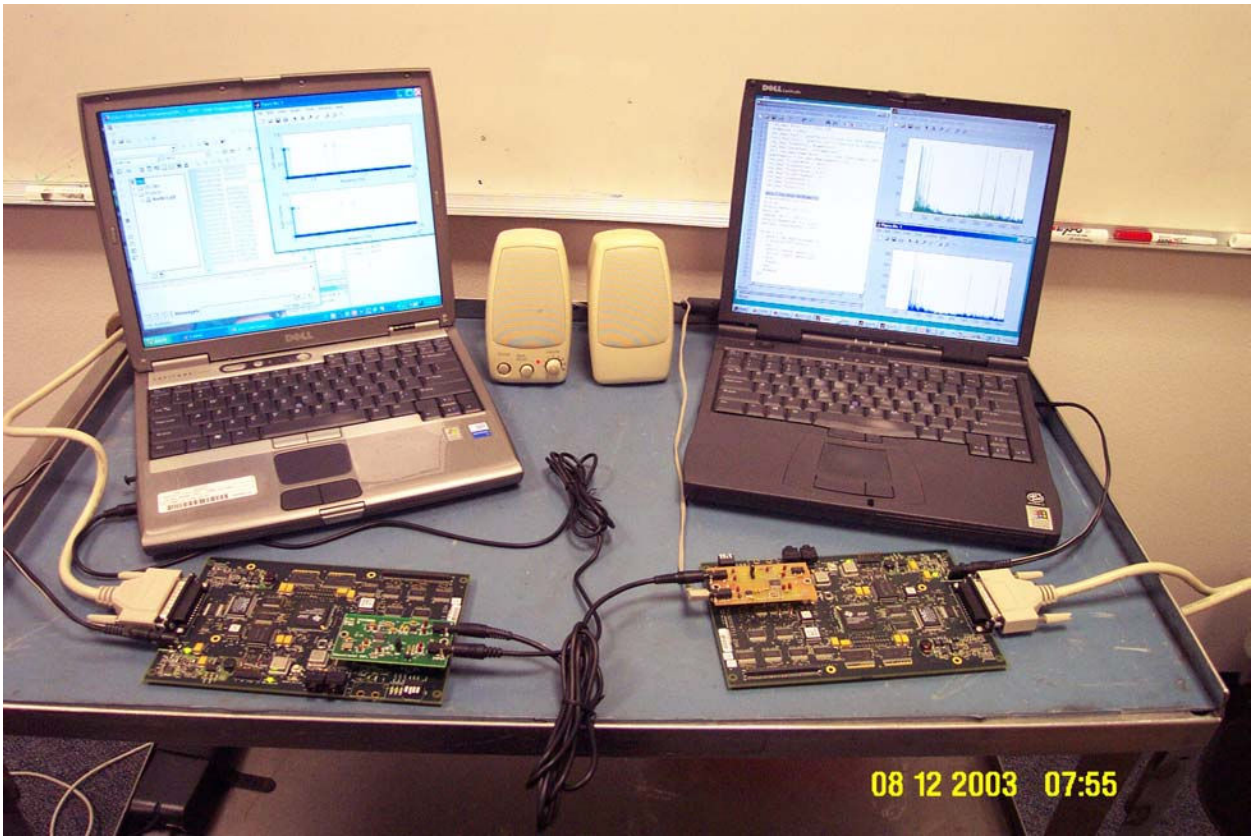

Figure 2. The laptop on the left loads and runs the voice removal C code on the left 6711 DSK and generates the source audio files from its CD player, while the right laptop running MATLAB serves as a spectrum analyzer along with the right 6711 DSK board that samples the output voice removed signal.

6. MATLAB to DSK Interface Software[5]

The interface between MATLAB and the DSK is encapsulated into a generalized interface command set that supports multiple input and output channels, variable sample rates, various triggering configurations, and variable frame sizes. The specific commands available are described in Morrow et al[5] and an example of its use in a real-time sonar application can be found in York et al[6]. The interface software only requires that the DSK tools be installed on the computer, and that the two files C6X_DAQ.DLL and DAQ_SIMUL.OUT be placed in a MATLAB-accessible directory. At the most basic level, this interface allows a novice user to operate the DSK as a data acquisition board with a simple command sequence, with no requirement to know how to use Code Composer, how to program in C, or any knowledge of how the hardware operates.

Initially all signal processing can be done in the MATLAB environment using "live" data acquired from the DSK. As the students progress, they can move processing functions from MATLAB down to the DSK by altering the DSK code (that was used to create the DAQ_SIMUL.OUT file), and still continue to use MATLAB as a graphical display engine.

Another use our students found for this interface is to make a simple spectrum analyzer to measure the performance of their "live" system, as shown in Figure 2. They hooked the output signal of the DSK running the voice removal algorithm into the input codec of another DSK, controlled by another PC running this MATLAB interface. With the data captured in MATLAB, they could easily take the FFT and quantitatively analyze the output of their system.

7. Classroom Uses of the System

After learning basic DSP theory and the concept of voice removal, our students were expected to meet the following specifications.

- Bass Separation Cutoff Frequency: 200 Hz
- Transition Bandwidth: 50 Hz (from 175 to 225 Hz)
- Maximum Passband Ripple (instruments signal and bass): 3 dB
- Minimum Stopband Attenuation (lead vocalist): 20 dB
- Sample Frequency: 48 kHz
- Linear phase filters are preferred.
- Choose a filter order within the processing capability of the C6711 DSP processor

The students are faced with several classic engineering design tradeoffs. Implementing all filters in Figure 1 using Equiriple FIR filters[8] results in four filters of an order of about 532. As each new left and right sample enters this system, all this processing must be completed and output before the next left and right samples arrive (20.8 microseconds at 48000 samples/second). Assuming the students use *unoptimized* floating point math in the C language on the C6711, they only have time to implement about one filter of order 540 [a hissing sound in the processed signal is a good indication if they exceeding the computation time per sample].

Thus they must either make justifiable engineering trade-offs *or* find faster implementations which still meet the specifications. Some ideas on faster implementations include

- Looking at the assumptions, it may be possible to reduce the system from 4 filters down to 2. With even more creativity, it may be possible to reduce to complete system down to one filter and one subtractor.
- Optimize the floating point code with pipelining[9]
- Switch from Floating point to Fixed point math[9]
- Switch from C to C with intrinsics or assembly language (usually beyond the capabilities of our students)

Fortunately, if the students figure out the solution using the first suggestion, greatly simplifying the block diagram in Figure 1, it is possible to meet all the requirements with unoptimized floating point C code on the 6711.

After designing their system, our students quantitatively test their MATLAB design using a file of test tones shown in figure 3. The tones common to the left and right channel above 200 Hz (simulating the lead vocalist) should be eliminated by 20 dB in the output signal. The remaining tones should be within 3 dB their original value. They then qualitatively analyze their MATLAB design by sampling a live recording and using the MATLAB-to-DSK interface discussed earlier, and listen to the quality of the processed signal.

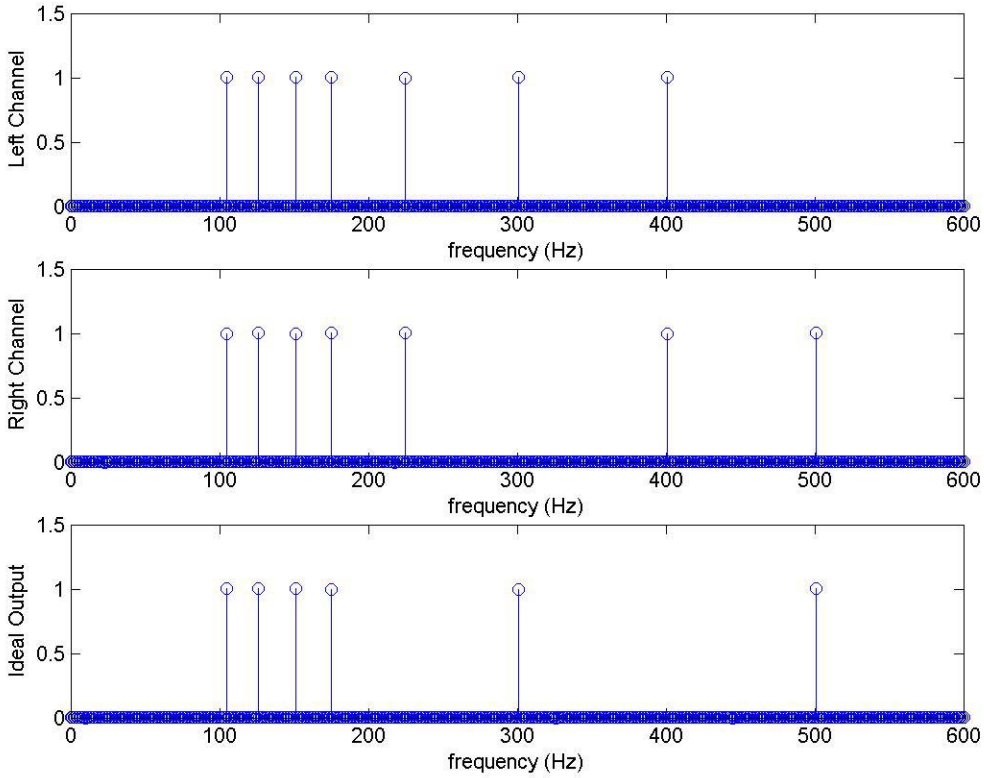

Figure 3. Spectrum of a left and right channel test signal. The ideal output plot illustrates the results after voice removal. Frequencies above 200 Hz shared by the left and right channel are eliminated.

In the second part of this exercise, they transition their MATLAB design to a C language implementation on the C6711 DSK board. They are provided with C source code already implementing the MATLAB-to-DSK interface to use as a guideline. They can implement their algorithm by simply modifying two functions controlling the codecs; the interrupt service routines that receive and transmit samples. If there is time in the course, we also introduce students to real-time programming issues, such as floating point versus fixed point, C versus assembly, SIMD operations, software pipelining, cache versus DMA data transfers, and circular and double buffering.[9]

Once the students have their real-time application working, modifying their design and hearing the real-time results has a tremendous (and enduring) impact on the students. They have considerable fun adding additional audio effects for incentive points, and some discover methods to recover the stereo signal after voice removal.

The last part of the exercise is a formal presentation of their design and implementation, in which they must justify the engineering tradeoffs they made in their system.

8. Results in the Classroom

We used this application as a final project for our senior-level introductory Digital Signal Processing course at the Air Force Academy for the first time in the Fall of 2003. We received very favorable feedback from the students. In addition to enjoying working on this application, many students indicated this project helped them really learn the basic DSP concepts, making abstract theory a reality to them. This project motivated some to want to take a follow-on course in DSP and to pursue a DSP application for their Senior Design projects.

9. Conclusions

We have developed an educational framework that will allow students to smoothly transition from their MATLAB design to a real-time DSP system implementation. This process allows real-world data to be gathered and used in the algorithm development while maintaining a link to MATLAB. The ease of using MATLAB was found to greatly reduce the frustration level of the students and shorten their development time. In addition, the DSP application of voice removal was found to be very motivational to the students and served to reinforce the basic concepts of DSP as well as teach practical implementation issues.

Bibliography

1. Weeder, T.J., "Build R-E's Vocal Stripper," in *Radio-Electronics,* Sept 1990.
2. You, C. & Sun H., "Multi-Band Adaptive Filtering Application On Vocal Mute," in *2002 6th International Conference on Signal Processing,* Aug 2002. pp 1711-14 vol 2.
3. Yoder, M.A., McClellan, J.H., & Schafer, R.W., "Experiences in teaching DSP first in the ECE curriculum," in *Proceedings of the 1997 ASEE Annual Conference,* June 1997. Paper 1220-06.
4. The MathWorks, Inc., Natick, MA, *MATLAB: The Language of Technical Computing,* 1999.
5. Morrow, M.G., Welch, T.B., & Wright, C.H.G., "An inexpensive software tool for teaching real-time DSP," in *Proceedings of the 1st IEEE DSP in Education Workshop,* (Hunt TX), IEEE Signal Processing Society, Oct 2000.
6. York, G. W. P., Morrow, M. G., Welch, T.B., and Wright, C. H. G., "Teaching Real-time Sonar With The C6711 DSK and MATLAB," *ASEE Computers in Education Journal*, Vol. 3, Sep 2002.
7. http://dspvillage.ti.com/
8. Ifeachor, E.C., & Jervis, B.W., *Digital Signal Processing: A Practical Approach,* Prentice-Hall, 2002.

GEORGE W.P. YORK, PhD, PE, is an Assistant Professor at the U.S. Air Force Academy. He received his PhD in EE from the University of Washington in 1999. His research interests include signal and image processing, embedded computer design, and ultrasound imaging. He is a member of ASEE, IEEE, and Tau Beta Pi.
Email: george.york@ieee.org

DANE F. FULLER is an Assistant Professor at the U.S. Air Force Academy. He received his Master of Science in Electrical Engineering from the Air Force Institute of Technology in 1997. His research interests include RADAR Target Identification and Engineering Education. Email: dane.fuller@usafa.af.mil

CHRISTOPHER M. RONDEAU, is currently an Electronics Engineer at Robins AFB, GA. He received his BSEE from the U.S. Air Force Academy in 2003. His research interests include digital signal processing with emphasis on audio and video applications. He is a member of IEEE. Email: christopher.rondeau@robins.af.mil