

MULTI-PLATFORM CBI TOOLS USING LINUX AND JAVA-BASED SOLUTIONS FOR DISTANCE LEARNING

E. K. Patterson, D. Wu, and J. N. Gowdy

Department of Electrical and Computer Engineering
Clemson University
Clemson, South Carolina, 29634, USA

ABSTRACT

This paper presents a set of software tools developed to enhance local and distance learning for speech and signal processing classes. The multi-platform paradigm is stressed in both computer-based instruction (CBI) [5] and World Wide Web (WWW), Java-based tools. The CBI speech tools are a package developed to enhance a graduate course in digital speech processing; the tools may be run on various platforms and will benefit local or distance learners taking the course via teleconferencing. Students may gain "hands-on" experience using this package on their own personal computer (PC) or a university computer. The WWW, Java-based tools have been designed to be used for interactive homework in signal processing classes but also have a wide range of applicability. JavaGram is the name for the currently developed application that allows students to easily "turn-in" homework diagrams, such as signal-flow charts, via the WWW. This is an improvement over past systems that use only forms in WWW homework for remote classes. These tools have been found to greatly enhance local and remote offerings of a course on speech signal processing.

1. INTRODUCTION

Traditionally, the engineering university environment involves a classroom experience of lectures given by faculty to students; a laboratory experience may follow in conjunct with the course, assuming sufficient funds or equipment are available that are applicable to the course. Because engineering courses are technically oriented, it is difficult to convey the large amount of information necessary in a classroom alone. Laboratory experience attempts to ameliorate this effort but may not be practical in certain cases. With these limitations, it may be challenging enough to efficiently conduct a local course. Today, however, classes are being offered remotely through university-and-business agreements to allow courses and continued job training where not previously available. The difficulties that are present in local engineering courses are magnified tremendously when the channel becomes some form of teleconferencing. The increased limitations on professor-student communication further decrease the flow of information and the student's opportunity to learn. Platform-independent and multi-platform tools for CBI and distance learning help improve courses taught both locally and remotely.

This paper discusses two efforts to improve remote and local courses. Both have been designed and used for a graduate course in digital speech processing but are applicable to any signal processing course and distance-learning in general. The first effort involves a set of multi-platform, CBI tools designed to provide a laboratory-like experience in conjunction with a speech processing course. The second is a platform-independent, Java tool for submitting and grading homework assignments, such as signal-flow diagrams, via the WWW. Together these tools expand the learning experience beyond the classroom, allowing a student more opportunity for interaction and, therefore, for learning.

The CBI speech tools are aimed at providing a laboratory-type experience to students in a first graduate speech processing class. The tools are a collection of modules that allow a student to experiment in a "hands-on" manner with concepts that are merely mathematical formulas in the classroom. Differences made by parameter changes may be seen or heard, rather than interpreted numerically. Such tools have become popular in recent years but are often available only on powerful, Unix™-based workstations. Our tools, likewise, were originally developed for Sun workstations. To truly aid students, however, easy accessibility outside of the classroom must be given. A crowded laboratory with restricted hours and competition from other students will not encourage a student to experiment beyond what is merely assigned. Because of this, the decision was made to port the tools to a more accessible platform, Linux. Today, PCs compete with workstations in power yet are less expensive and more likely to be owned by students. Linux is an increasingly popular platform that brings the power of Unix™ to the PC. With the speech tools ported to run on a Linux platform, students may run the tools at home on their own machines where they are more likely to experiment and gain from the "hands-on" experience. The cost of creating a dedicated laboratory of Linux workstations for the speech tools and even for research-related software such as Matlab™, HTK™, or ESPS/waves™ is also significantly less costly. The speech tools are simple to install and use, can be made easily accessible, and have been found to aid both local and remote classes.

Distance-learning is increasing in popularity and usefulness because of modern communication tools such as teleconferencing and the Internet. Communication is never as effective, however, when it is not direct, person-to-person. Communication schemes not only for teleconferenced lectures but also for question, homework, and other extra-classroom communication make distance-learning more difficult and

restrictive in many ways. Currently, one of the most deficient areas is communication of homework. A remote student does not have the option of “handing-in” an assignment but must result to phone, fax, or more recently HTML forms. These methods tend to be slow and restrictive. JavaGram is a platform-independent tool that demonstrates the capabilities of Java for providing greater ease and more interaction with distance homework. It provides a simple and quick interface for submitting and grading assignments and also is a good example of the power that is available for developing further educational tools using the WWW.

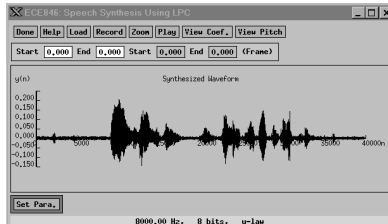


Figure 1. Example of a speech tool module for Linux.

2. SPEECH TOOLS

The CBI speech tools were developed for a graduate course in speech processing but could be applied in many signal processing courses. The purpose of the tools is to provide students a laboratory-like experience that may not otherwise be available. The tools are made up of several modules that provide examples and opportunity for experimentation with phoneme segmentation, vocal tract modeling, speech coding, speech synthesis, and speech recognition. They are designed to be used in assignments that parallel a speech-processing course as taught from [3]. Students may also experiment on their own to gain enhanced course experience. The tools have recently been ported to Linux to increase accessibility with the multi-platform and personal CBI paradigms in mind [5].

2.1 Speech tool modules

The speech tools consist of five main utilities that parallel a course in speech processing: phoneme segmentation, vocal tract modeling, speech coding, speech synthesis, and speech recognition. A main menu allows a student to choose which module to run. In the class, several assignments are given to introduce each module and demonstrate the concepts behind it. After being introduced to each, a student may experiment as desired to reinforce course material.

The first module seeks to demonstrate phoneme segmentation. Speech segments several seconds long may be loaded and displayed as acoustic waveforms. The capability to record, load, pan, and zoom across speech waveforms allows a student to become comfortable with the characteristics of speech, recognizing features such as silence, voicing, and noise. Sections of the waveform may be selected on-screen and played through the computer’s audio system. A typical assignment is phonetic transcription of a sentence of speech. By loading and displaying the waveform of a spoken sentence, a student may

proceed to select and play segments until familiar with general phonemic boundaries. The fact that phonemes are very transitional is also demonstrated. After experimenting with this module, the student should have a feeling for general characteristics of speech waveforms and should be more fluent with phonemic transcription.

The speech production model is the second speech tool module. Consisting of a modifiable, block diagram that demonstrates excitation through vocal tract and radiation models, this module allows a student to study one of the most popular models for speech production. Given control of such parameters as excitation, tube size, and tube number lends insight into what is merely a mathematical model in the classroom. Students may hear changes in produced sounds based on the parameter changes made.

A third module demonstrates the basics of speech coding. Students may change parameters such as the order and number of bits in simple coding schemes: PCM, μ -law, APCM, and ADPCM. Again, by making parameter changes, a student may visually note the changes made in the decoded waveform and also audibly evaluate the differences in decoded speech.

The fourth module gives a basic demonstration of speech synthesis through use of linear predictive coding (LPC). The student may change the order and other parameters of an LPC model and see and hear the effects of the changes in synthesized speech. The student may also see the LPC coefficients produced by the model.

The final module allows experimentation with speech recognition. This module is designed to be used with a class project to create a simple recognizer. The module appears as a block diagram of the basic steps of common recognizers: preprocessing, feature extraction, pattern matching, template creation, and recognition decision. Each of the blocks allows a student to test a simple demonstration example that comes with the tools or link in a student’s own program to perform the particular step of the recognizer. As the project proceeds, a student should be able to incrementally test all parts of a simple, created recognizer by linking in the parts as they are developed.

2.2 Linux and personal CBI

The speech tools were created two years ago to run on Sun workstations that are a portion of computer facilities widely available to engineering students at the university. Workstations are accessible for students while on campus but laboratories are sometimes crowded. Because of the incompatible platforms, the software could not be run at home where experimentation would likely be done and enjoyed. During remote teaching to other universities and businesses, the tools proved to be powerful aids for the course, but finding compatible workstations and conducting remote installations proved difficult. To make the speech tools a more effective software package, they needed to be made available as personally owned and maintained software, the essence of CBI [5]. The multi-platform paradigm works well in implementing personal CBI. Software that is more readily available and available for home use encourages student motivation and understanding.

As PCs become more powerful, popular, and less costly, they are beginning to replace many of the tasks for which workstations have been used. Engineering students are also highly likely to own their own PCs. Because of this, the personal CBI concept is becoming a more practical and important idea. Instead of competing for institutional resources, students may build their own collection of tools that may be applied in many courses. Because much of scientific software is run on the Unix™ platform, Linux can be used as a very important tool in the personal CBI paradigm.

Linux is a relatively new operating system, first available on the Internet in 1991 [1]. It is a freely available Unix™-based operating system for PCs (and a growing number of other architectures). Because Linux is distributed (and developed) openly, it is widely available and easy to obtain; it may be downloaded from various Internet sites freely or purchased inexpensively on CD-ROM. This is very much in the spirit of personal CBI; it is important to keep students' costs low so as to encourage involvement and allow a library of CBI tools to be built. Many powerful, scientific applications are available commercially and freely for Linux. Matlab™, one of the most popular software tools in keeping with the personal CBI paradigm, is an example of the many commercial titles available for Linux. There is also a myriad of freely available, yet very powerful scientific software for Linux available on the Internet. Many more software titles that have traditionally been available for Unix™ are being ported to Linux. Also, modern PCs running Linux compete with and can surpass workstations in speed and flexibility. Finally, Linux is a very powerful development platform, with strong C/C++ and other language development tools. Because of the availability, power, and software base for Linux, it makes an excellent operating system for the personal CBI paradigm.

The porting of Unix™-based software to Linux is usually a simple process, consisting mainly of recompiling the source code on the Linux platform. Various hardware specific issues and minor differences in libraries, such as for window-managers may have to be dealt with for a successful port. Addition of new drivers and kernel improvements are making Linux even more powerful and an easier system to which to port.

Because of the power and relative ease, the speech tools were ported to Linux. Installation is now simpler and available to nearly any 486-based or better PC with sound capabilities. Students may use the software at home, and inexpensive, Linux-based workstations available on campus have been setup with the software. Personal installation of Linux is encouraged and helps provide the student with more computing experience. A good platform for building further personal CBI tools is also established.

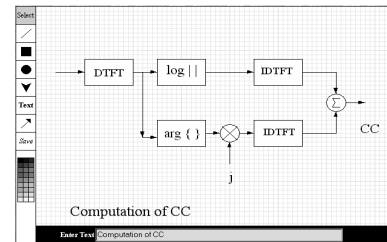


Figure 2. Example of a student using JavaGram.

3. JAVAGRAM

The second thrust of the project for improving the teleconferenced version of the speech processing course involves JavaGram (<http://moo.eng.clemson.edu/javagram>). JavaGram is an object-oriented drawing package, written in Java, that is runnable over the Internet via a web browser such as Netscape™ or Internet Explorer™. Diagrams may be created quickly and easily with JavaGram and saved to a university server. From the server the instructor or grader may easily access and grade the assignments, also via a web browser.

Teleconferenced courses are becoming much more popular because of many newly available communication tools. One of the important tools for recent distance-learning has been the Internet. WWW pages are being used instead of mailing and faxing for professor-student communication outside of class time. Forms and other CGI and script-based pages have been used for greater speed and flexibility over traditional communications. Interaction, however, is often limited and only text-based. JavaGram is an attempt to give more flexibility to WWW-based homework and demonstrate further capabilities of Java that may be used for future distance-learning software.

3.1 JavaGram Capabilities

The nearly platform-independent language Java is a powerful tool for adding interaction to WWW pages. This power can be applied in many ways to distance-learning solutions. Many applets are available that demonstrate the capabilities of Java, but until recently most have been merely demonstration “toys.” The newness of the language is wearing off, however, and some powerful applets and applications are being developed. JavaGram shows some of the power Java gives and is a good tool to be used with teaching courses remotely. As the multi-platform capabilities of the speech tools software increased their accessibility and usefulness, the platform-independent paradigm makes applications developed in Java, such as JavaGram, very easily accessible for students. Some differences in Java Virtual Machines between browsers and architectures did present difficulties during development; however, newer releases of Java may solve some of these problems.

JavaGram is a diagram-drawing package that allows students to easily create homework diagrams, such as signal-flow charts, merely using a web browser. Accessible from any Internet-connected PC, a student may work on and submit graphical homework assignments. The package itself allows creation of graphic objects, such as lines, arrows, boxes, and text. The

objects may be resized, relocated, copied, or deleted. New graphical objects may be added to the code with relative ease. This makes JavaGram very flexible and expandable.

Java applets are protected from saving files on client machines from which web browsers are run. Ideally, files should be saved on the server, anyway, to also allow easy access for the grader. Because of this, JavaGram uses a package freely available on the web (<http://www.ncs.com.sg/java/jfs/>) called the Java File System (JFS). JFS is a powerful tool that allows client-server communication from within Java applets. It provides a set of Unix™-like file system tools to Java applets.

Students are given accounts on a server and login with a userid and password when the JavaGram web page is accessed. The JavaGram applet then begins and gives the student various object-based drawing tools to complete a diagram. The student may then save the diagram back to the server by merely pressing the save button. The GIF format is used because it greatly compresses the image for transmission over the Internet but is not computationally intensive. Other formats such as JPEG require more computation, and Java applets still tend to be relatively slow, although this is improving.

Once a student saves a diagram, it is accessible by the grader. A JavaScript page allows a grader to easily view students' homework diagrams via a web browser. With this combination homework may be completed and graded from nearly any Internet-connected PC. This provides a powerful and flexible paradigm for remote course homework and helps solve difficulties associated with faxing, mailing, or phoning assignments.

3.2 Further possibilities

JavaGram is a fairly flexible tool for conducting homework assignments via the Internet. Perhaps even more importantly, it demonstrates some of the power that is now available using Java and the WWW. Many other client-server applets could be created using JFS or other code to create a rich base of applications to be used with distance-learning. Further interaction will aid professor-student communication and improve the distance-learning experience in general.

JavaGram is being used and tested currently in courses at Clemson University. Additional features are being considered for inclusion in enhanced versions of JavaGram. Increased and improved capability for expressing mathematical formulas in homework is an important area for signal-processing courses taught remotely. Other possibilities for enhancement include increased interaction such as active system simulation of signal flow diagrams. The object-oriented paradigm allows possibilities such as expanding current arrow and box objects into system objects that could simulate simple system properties. As an example, an excitation box may be connected to a filter box to an output box that shows the system output. These are a few of many possibilities that could be implemented via Java and the WWW.

4. CONCLUSIONS

The framework of the engineering classroom is changing. Multimedia and personal CBI paradigms are increasing student interaction and enhancing the traditional lecture experience. Communication tools are increasing the effectiveness of distance-learning, making it a viable tool for expanded course offerings among universities and businesses. Increasing interaction and motivation of students involved with local and remote courses is one of the main areas that needs to be addressed for making these new learning-experiences useful.

Personal CBI tools can be very important for enhancing the classroom experience. The developed speech tools have proved very useful in conducting a speech processing course locally and remotely. Stressing multi-platform capability and easy accessibility with Linux has improved their effectiveness. Linux is a very capable tool that is in the spirit of personal CBI and expands a student PC's powers. As demonstrated by the success of the speech tools, Linux makes an excellent platform for CBI in general. Encouraging a student to experiment personally motivates learning and evokes enthusiasm beyond that garnered by the traditional classroom experience.

Also stressing accessibility and increased student interaction, JavaGram demonstrates the powerful capabilities of Java and the WWW, especially as used in remote courses. Increasing the extra-classroom experience beyond HTML forms or faxes greatly improves distance-learning. The capabilities of JavaGram are growing and yield more power and ease for the student and graders involved in a remote course. Both the speech tools and JavaGram have yielded successful additions to the speech processing course, as a local and remote class. Multi-platform and personal CBI tools are both powerful opportunities for enhancing the traditional engineering classroom experience.

5. REFERENCES

- [1] Beck M., Bohme H., Dziadzka M., Kunitz U., Magnus R., and Verworner D. *Linux Kernel Internals*. Addison-Wesley, Harlow, England, 1996.
- [2] Chatfield S., Cochran D., Sadaka M, and Sinno D. "A System Characterization/Identification Laboratory Teaching Tool for Internet". *IEEE International Conference on Acoustics, Speech, and Signal Processing*. Atlanta, May 1995.
- [3] Deller, J., Proakis J., and Hansen J., *Discrete-Time Processing of Speech Signals*. Macmillan Publishing Company, New York, NY, 1993.
- [4] McClellan J., Shafer R., Schodorf J., and Yoder M. "Multi-Media and World Wide Web Resources for Teaching DSP". *IEEE International Conference on Acoustics, Speech, and Signal Processing*. Atlanta, May 1995.
- [5] Taylor F., Mellott J., and Lewis M. "Spectra – a Hands-On DSP Learning Experience". *IEEE International Conference on Acoustics, Speech, and Signal Processing*. Atlanta, May 1995.