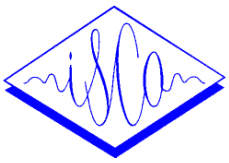

2nd International Conference on
Spoken Language Processing (ICSLP 92)
Banff, Alberta, Canada
October 12-16, 1992

# AN INTERACTIVE ENVIRONMENT FOR SPEECH RECOGNITION RESEARCH

*Mark Fanty, John Pochmara and Ron Cole*

Center for Spoken Language Understanding
Oregon Graduate Institute
Beaverton, Oregon, 97006

## ABSTRACT

A UNIX software environment for speech recognition research is described. This environment is undergoing development under an NSF Software Capitalization grant and will be made public when complete. The speech tools will allow users to compute and display a variety of signal representations of speech, to label speech at a number of levels, to train and evaluate neural network classifiers and to display the processing stages of speech algorithms. The tools include file formats and support routines for audio files, two-dimensional data files (e.g. FFT output), neural network classifiers and time-aligned label files. The LYRE and AUTOLYRE programs display speech data in a variety of ways. NOPT is a conjugate gradient descent neural network training program which is limited in flexibility but easy to use and fast. A distributed version for networked workstations exists and will continue to be enhanced. A small but useful set of signal processing routines is provided including Perceptual Linear Predictive Analysis (PLP). Speech algorithms include a pitch tracker for high-quality speech and dynamic programming optimization code for use with phoneme probability matrices (e.g. as computed by a neural network classifier). Although the software currently reflects the biases of the Center for Spoken Language Understanding towards neural-network-based recognition using DFT- and PLP-based features, we hope that the user community continues to enhance the tool set. We have tried to design it with flexibility and growth in mind.

10.21437/ICSLP.1992-200

## INTRODUCTION

The Center for Spoken Language Understanding is developing the software tools used internally for public distribution. Our hope is to enable research for those unable to afford commercial packages and to encourage the sharing of speech processing software by providing a common foundation. The tools are written in C and Perl. The display routines use X windows. They consist of routines to manipulate and display basic data structures (e.g. waveforms and spectrograms), some signal processing routines, a neural network training program, and some utilities.

We anticipate public release of the tools sometime during the Fall of 1992. They will be made available for ftp. While we cannot provide support, we will provide updates and bug fixes when they are available. We also plan to create a mailing list so users can directly communicate with each other. If the user community enhances and expands the tool set, it could grow into a significant resource.

## FILE FORMATS

### Audio Files.

Of the many audio file formats which exist, only three are supported: *adc* developed by Carnegie Mellon University and used previously at OGI; SPHERE-headered waveform files as distributed by NIST (including MIT enhancements to compress the data without loss of information); and headerless mu-law data (as produced by Sun SPARCstations and DECaudio). It is our hope that a good many more formats will be added to this list.

All applications which read audio files use the routine *AudioRead* which returns 16 bit linear data and the sample rate in samples per second. By enhancing the *AudioRead* routine to recognize and read a new format (converting to 16 bit linear samples), all tools are able to read the new format as well. Each audio format also has low-level routines which may be able to access more information depending on the format. These routines are available to the user.

### Two-dimensional Data.

The *tdat* file format represents two-dimensional data with column values representing time and row values free to represent any numerical value. It consists of a variable-length ascii header ending with the string "END OF HEADER." The data follows the header and is binary with big-endian (Sun) byte order. The user is free to define new header fields—of course the standard tools will ignore them. Certain fields are expected, such as the number of columns and rows, the type of data, and the row labels. Row labels are ascii strings describing each row (e.g. frequency values for DFT data; phoneme names for a phoneme probability matrix) and are displayed by the LYRE display program (describe below) when the pointer enters that row in the *tdat* display.

### Label Files.

Label files provide time-aligned ascii labels and are primarily used by us to label word and phoneme boundaries. We wanted to use the same format as the TIMIT database, which is one ascii line per label with three fields: beginning sample point, ending sample point and label. However, without knowing the sampling rate, this file cannot be correctly displayed. We opted to add a one-line ascii header which gives the size of each time unit in milliseconds.

### Neural Network Files

The neural network file format is for the NOPT neural network optimization program described below. It is not a general purpose network file format. It has been changed from the old format [1] primarily in that the byte order is now fixed. The format is for feed-forward networks with total connectivity between adjacent layers. All weight values and activations are stored, as is the training iteration count.

## THE LYRE DISPLAY PROGRAM

In a sense, LYRE is the heart of the tool kit. It is an X window display program which can simultaneously display any number of waveform, *tdat* and label files. The displays are aligned

so that the center of each window corresponds to the same time. The resolution of each window can be individually set. When the pointer is in any display, a vertical line is drawn in all displays at the corresponding time.

Each window has its own key and button functions. Part or all of a waveform window can be played. (The samples are written to disk and a standard play function is called.) Contiguous segments can be written to disk. No editing or splicing is possible. This functionality will probably not be added to LYRE since it is inconsistent with the goal of several displays of the same speech.

*Tdat* displays give a readout of the row label as the pointer is moved. For example, when pointing to a formant in a spectrogram, the frequency can be read. The data display is separated into positive and negative values. For each, a minimum and maximum data value is specified. These are set automatically to the min and max in the data at read time, but can be reset to any value manually. A min and max color and the number of discrete color values are also specified. For example, if the positive min is 0.0 and the max is 0.6; if the min color is white and max color is red and if the number of colors is 30, then the data values between 0.0 and 0.6 will be represented by 30 increasingly saturated shades of red with 0.0 being white and any value larger than 0.6 being red. The shape of the mapping is controlled by the exponent parameter, with 1.0 producing a linear mapping. No attempt is made to smooth the display. The data appear as rectangular boxes of a uniform color. While this is less aesthetically pleasing than a smoothed display in some instances (e.g. spectrograms), it more accurately depicts the data and is desirable for other displays (e.g. matrices of phoneme probabilities).

The label display draws lines for the left and right boundaries of a region and displays the label between those lines. The text corresponding to each label can be edited. New regions can be created and old ones destroyed. The labels can be read from and written to a label file, with the option to "write-aligned" which moves all right boundaries over to the next left boundary. This option is a big time saver when labeling speech files. Text is entered in the box which has the current focus (indicated by dotted boundaries). Entering return moves the focus to the next box to the right. While it is possible to have overlapping boxes, the display is not appropriate for hierarchies of labels.

Files are explicitly loaded into LYRE by entering their names. A variant of LYRE called AUTOLYRE is created with a list of file names and display types to monitor. AUTOLYRE checks the modification times of these files a few times every second and will automatically redisplay any file which has been modified. This mechanism is extremely convenient for creating display backends for speech demos, or for any task which is repeated for a large number of files (such as labelling and chopping off excess silence). Figure 1 shows a screen dump of AUTOLYRE.

A simple example will illustrate the use of AUTOLYRE. The following csh script starts an AUTOLYRE with a single waveform display corresponding to the name "XX.wav." The script will loop over all ".wav" files in the current directory, displaying each file in turn. The user can chop off silence before and after the speech by using the left and right mouse buttons to define a segment, then writing it (to the default name which is the original file name with a "-t" added). The script works by sequentially linking each file to the name "XX.wav". If the user writes a segment and creates a "XX-t.wav" file, the script copies it to a permanent name which is the original name with a ".cwav" suffix instead of a ".wav" suffix.

```
#!/bin/csh
autolyre -basename XX -config W100.wav &
foreach wfile (*.wav)
  rm -f XX.wav
  ln -s $wfile XX.wav
```

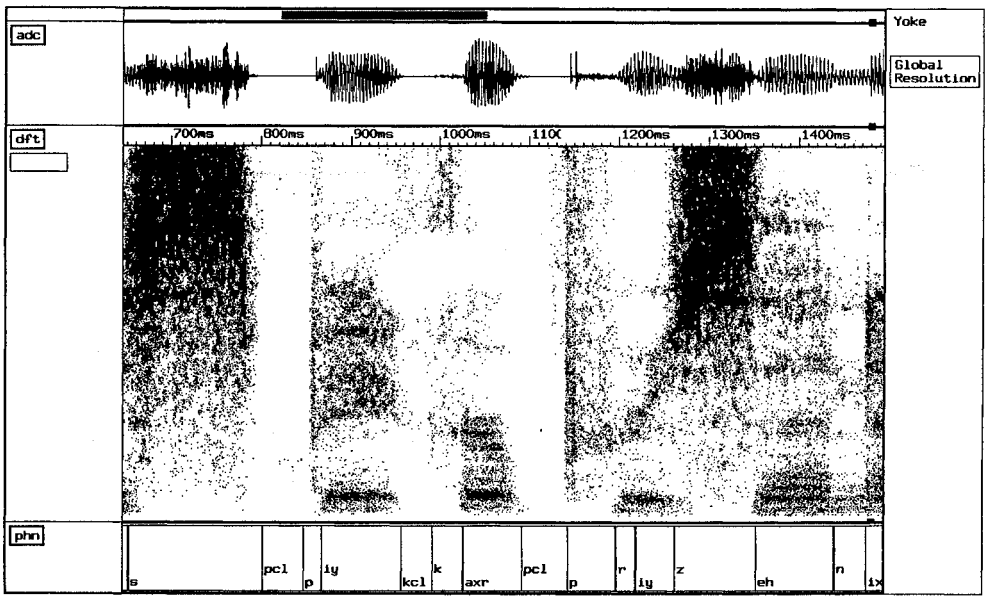


*Fig. 1 – Screendump of auto_lyre with waveform, B&W spectrogram (tdat) and label windows for a segment of speech from the TIMIT database.*

```
  echo -n "write chopped file and hit return "
  set ans = $<
  if (-e XX-t.wav) mv XX-t.wav ${wfile:r}.cwav
end
rm -f XX.wav
```

## NEURAL NETWORK CLASSIFIER

### NOPT

An earlier version of NOPT is described in [1]. NOPT trains neural network classifiers using back propagation with conjugate gradient optimization. The free parameters are the number of layers, number of units in each layer and random seed for defining initial weights. Adjacent layers are totally connected. There is no bias input, but there is an extra input in the first layer with a constant value of 1.0. The input data is specified with an ascii file which represents each pattern as an integer class followed by the input values for the first layer. The network is always trained with each input vector belonging to exactly one output class. The desired outputs for the correct and incorrect classes is user-settable.

We have a simple distributed version of NOPT which divides the data and runs on any number of idle workstations. Currently the data is divided and each process is started manually, but further work is planned.

NOPT is inflexible, but fast and simple to use. While we may invest some effort making it more general, we would prefer to take advantage of the many public domain programs already available.

### NOPT Utilities

NOPTEST will evaluate a data file using a trained weight file and produce a confusion matrix. NOPTRUN will run NOPT and call NOPTEST on the series of weight files written by NOPT as it trained. It summaries the performance of each saved iteration in a table. NOPTREAD will produce an ascii printout of a weightfile.

### XOPT

XOPT is an interactive graphics tool which displays the weights in a weightfile. The nodes in the network are represented by icons in a window. Selecting a unit with the middle mouse button displays it's receptive field by resizing the icons representing nodes in the previous layer proportional to the weight to the selected unit. The range of display sizes are determined dynamically based on the standard deviation of weight values between the layers. Similarly, the projection of a node's activity onto the subsequent layer can be displayed by selecting that unit with the left mouse button.

The right mouse button can be used to get some idea of the net influence of a unit on another unit more than one connection away. Selecting an output unit in a three-layer network will project activity backwards over all paths to the input layer and size the inputs accordingly. The weight on a path of connections is the product of the individual weights. For example, a weight of 0.6 from input unit 3 to hidden unit 2 and a weight of 0.2 from hidden unit 2 to output unit 4 is treated as a weight of 0.12 from input unit 3 to output unit 4.

An input vector file can be loaded into XOPT and the pattern of activation resulting from each vector observed. Given a particular input, errors of plus and minus one and be defined on the output using the mouse and these errors back propagated to the input layer. Icon size reflects the gradient of the input activation.

The location of the nodes in the display is user-settable via a simple layout language which includes commands for placing text in the display window.

## SIGNAL PROCESSING ROUTINES

All routines are written in C. The source for each is provided in the documentation.

### FFT

Several interfaces to the FFT are provided. There is a simple FFT routine which is passed a power-of-two length complex vector and returns the same. There is an FFT routine optimized for real input data. There are windowing fuctions to apply before calling the FFT. Most of the time, the *ComputeFFT* function provides the most convenient interface. It is passed audio data and computes the FFT for a number of windowed frames at equal increments, returning the output in a two-dimensional data structure. The user can specify the length of the analysis window, the frame rate and other parameters. Like many other functions, this routine is available from the command line as well.

### PLP

The Perceptual Linear Predictive (PLP) speech analysis technique [2] estimates an *all-pole* (autoregressive) model of the *auditory-like* short-term speech spectrum. PLP has been shown to be efficient in suppressing speaker-dependent components in the speech signal. Compared to the conventional linear predictive (LP) analysis of speech (which estimates an all-pole model of the short-term *power* spectrum of speech), the PLP model order in speaker-independent applications is typically lower. Computationally, PLP is at least as efficient as the conventional LP analysis. Preliminary experiments at OGI show 8 PLP coefficients to be as effective as 40 or more DFT coefficients when input to neural network classifiers [3, 4].

RASTA PLP [5] is a modification of PLP in which each frequency channel is bandpass filtered in the log domain. This has the effect of normalizing for input channel (e.g. it can make a huge difference in the performance of the system when using a new kind of microphone.) As of this writing, all working speech systems at OGI use either PLP or RASTA PLP.

### Cochleagram

The cochlear model designed by Lyon [6] and described by Slaney [7] converts a sound waveform into a multidimensional vector that represents the information sent from the ear to the brain. The cochlear model does not try to accurately model the internal structure of the ear but only to approximate the information contained in the auditory nerve.

### FIR Filters

A subroutine to apply FIR filters to audio files is provided as well as a selection of filters. We hope to add a C implementation of the McClelland-Parks filter design program as well [8].

### Autocorrelation

A subroutine to compute the autocorrelation of a audio file is provided.

## SPEECH ALGORITHMS

At least a couple of useful speech algorithms developed by the Center will be included: a neural-network based pitch tracker for high-quality speech [9] and a dynamic-programming search for

use with neural network (or other) phoneme probability matrices. The input to the search is a matrix of phoneme scores over time, word models defining the pronunciation of words in terms of these phonemes, and a finite-state grammar defining the allowed sequences of words. The output is the word sequence with the highest score.

## UTILITY ROUTINES

For each audio format supported, routines exist to convert it to every other supported audio format when possible (e.g. 16kHz data can currently be converted to 8kHz mu-law, but 10kHz data cannot). Audio and *tdat* files can be beheaded or converted to an ascii stream. This makes it easy to use these files with general purpose UNIX utilities which don't understand the headers.

Autogain will modify a audio file so that its maximum amplitude is in some predefined range. This is especially useful when listening with headphones.

## FINDPHONE

FINDPHONE selects requested phonemes out of a labelled database. Given a database of label files and corresponding waveform files, the BUILDINDEX program makes an index of every label and its immediately surrounding labels. The index is used by FINDPHONE to construct new waveform files which are the concatenation of phonemes selected at random (or in sequence) from the larger database. Used with AUTOLYRE, FINDPHONE is an excellent browsing tool. For example,

```
findphone timit -count 30 -pre  aa ao, l, -
```

will select 30 instances of /l/ from TIMIT for which the preceding phoneme is either /aa/ or /ao/ and construct a single new waveform which is the concatenation of these 30 phoneme pairs separated by silence. The "-" is a wildcard for the phoneme following the /l/. The "-pre" means save the phoneme preceding the /l/. Once the new waveform is created, a spectrogram can be computed and both the waveform and spectrogram displayed in AUTOLYRE automatically. With a little wrap-around csh script, the user can repeatedly type patterns and in a few seconds see (and hear) several examples.

## AUDIO OUTPUT

One of the most troublesome features of our old environment was correctly producing audio output. We have a heterogenous environment with at least six ways of playing audio. The new mechanism, which is still under development as of this writing, will work as follows. While each device has specific software, all programs, scripts, users call "audioplay" which can read all supported audio files and play them on the correct device as determined either by default from the X "DISPLAY" variable and a table, or as set by the user directly.

For MacIntosh, we acquired a network play program from Apple which runs an audio server (TCPPLAY) on the MacIntosh and uses the UNIX program RPLAY to send audio over the ethernet to TCPPLAY.

For SPARCstations, we have an audio output program which can use either the speaker or the external jack. This is not yet running over the network. For DECstations, we use DEC's audio server, which we are not free to redistribute. We also have appropriate output programs which use Gradient Desklab, also not yet made into network servers.

## DOCUMENTATION

One of the major goals of this project was to document the wide variety of speech algorithms we were using. Every program, function and file format will have a man page. In addition, we are developing a user's manual to provide an introduction and overview, including examples.

## DISTRIBUTION

Release is expected sometime during the Fall of 1992. The tools will be freely available via ftp. Enquires can be emailed to "tools@speech.cse.ogi.edu" or write to the Center.

## ACKNOWLEDGEMENTS

This research was supported by an NSF grant (IRI-9020571) and by an equipment donation from Digital Equipment Corporation. LYRE and AUTOLYRE were originally written by Fil Alleva of Carnegie Mellon University. Fil has contributed in many other ways to the speech environment at OGI. Findphone is a generalization and re-implementation in perl of the fastphone program developed by Aaron Hillegass and Robert Benedetto at MITRE Corp. PLP and RASTA-PLP code was provided by Hynek Hermansky of U.S. West. The cochleagram code is from Malcolm Slaney of Apple Computer Inc. Lossless waveform compression code is from Mike Phillips and Peter Daly of the Spoken Language Systems Group at MIT.

## BIBLIOGRAPHY

[1] Barnard, E. and R. A. Cole, "A neural-net training program based on conjugate-gradient optimization." Technical Report No. CS/E 89-014, Oregon Graduate Institute, 1989.

[1] Hermansky, H., "Perceptual Linear Predictive (PLP) Analysis of Speech," *J. Acoust. Soc. Am.*, **87**, pp. 1738–1752, 1990.

[3] Fanty, M., Cole, R., and M. Slaney, "A comparison of DFT, PLP, and Cochleagram for alphabet recognition," *Proceedings of the Twenty-Fifth Asilomar Conference on Signals, Systems and Computers*, November, 1991.

[4] Creekmore, J. W., Fanty, M., and R. A. Cole, "A comparative study of five spectral representations for speaker-independent phonetic recognition," *Proceedings of the Twenty-Fifth Asilomar Conference on Signals, Systems and Computers*, November, 1991.

[5] Hermansky, H., Morgan, N., Bayya, A. and P. Kohn, "RASTA-PLP speech analysis technique," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, San Francisco CA, March 1992, pp. 121-124.

[6] Lyon, R. F., "A computational model of filtering, detection, and compression in the cochlea," *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, 1982.

[7] Slaney, M., "Lyon's Cochlear Model," Apple Technical Report #3, Apple Computer Inc., 1988.

[8] McClellan, J. H., Parks, T. W. and L. R. Rabiner, "A computer program for designing optimum FIR linear phase digital filters," *IEEE Transactions on Audio and Electroacoustics*, AU-21, N. 6, pp. 506–526, 1973.

[9] Barnard, E., Cole, R. A., Vea, M. and F. Alleva, "Pitch detection with a neural net classifier," *IEEE Transactions ASSP*, **39**, pp. 298–307, 1991.