

# SIGNAL PROCESSING TOOLS FOR SPEECH RECOGNITION<sup>1</sup>

*Hualin Gao, Julie Baca and Joseph Picone*

Institute for Signal and Information Processing  
Mississippi State University, Mississippi State, MS 39762 USA  
{gao, baca, picone}@isip.msstate.edu

## ABSTRACT

Signal processing tools used to extract feature vectors from speech signal were developed by the Institute for Signal and Information Processing. The tools were designed to process general-purpose DSP functionalities and to act as parts of core components of ISIP speech recognition system. The design goal of the tools is to simplify the software development for researcher in both academic and industrial area by simply drawing block diagram to expression design ideas without any programming. This paper discusses the issues related to the design and implementation of such digital signal processing tools.

## 1. INTRODUCTION

Signal processing tools of a speech recognition system extract feature vectors from the speech data. It is generally referred to as a front end tools. There are a lot of signal processing toolkits available, Matlab is one of the most popular used tools in the signal processing area. Those toolkits really provide powerful computation and analysis ability for users, especially the graphical ability, but the major problems are: First, all those need more or less programming if users want to evaluate new ideas using existing algorithms. Second, if users want to add new algorithms to existing system, the base code of the existing system will need to be modified, these will need a lot of programming and debugging. This will slow down many research efforts. Third, any change over existing computation procedure will need to re-debug the existing procedure. Fourth, there are other two special problems for speech recognition front end. One is synchronization along the data flow graph if there exist two or more paths using different numbers of algorithms to reach a same point. The other is if the algorithm needs more than one frame data such as window and calculus, the data preparation for multiple frames data for those algorithms is a very important step. In order to overcome all those problems and to provide

users a powerful, flexible and no programming tool for signal processing, ISIP has focused on the development of a modular and flexible environment of signal processing tools. The key differentiating characteristics of this system include [1]:

- competitive technology with maximum flexibility;
- unrestricted access via the Internet;
- well-documented APIs to facilitate new programming;
- an object-oriented software design.

This paper discusses GUI-based tools that enable users to implement front ends by drawing block diagrams of signal processing functions without any programming. The tools are parts of core components of the ISIP speech recognition system. A brief description of low-level programming interfaces between signal processing tools and speech system is also introduced later.

## 2. GUI-BASED SIGNAL PROCESSING TOOLS

Signal processing tools of the extract feature vectors from the speech data. The goal of developing new signal processing tools is to allow a wide selection of algorithms, to vary every parameter of each algorithm easily and to provide users an efficient environment for the evaluation of new research ideas.

The design requirements for these tools included:

- a library of standard DSP algorithms to provide basic DSP functions;
- a block diagram approach to describing algorithms to realize the rapid prototyping without programming;
- an ability to plug in new algorithm classes and functions.

The first requirement will provide a basic DSP tool to directly realize a single algorithm such as windows, filters or energy, with simple programming by building a algorithm object and calling its functions. This will also enable users to employ these fully tested standard DSP algorithm in their own software.

The second requirement will provide a powerful tool to directly realize a rapid prototyping of new ideas by using a single algorithm or combining existing algorithms and their

---

1. This material is based upon work supported by the National Science Foundation under Grant No. EIA-9809300. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

data flow through block diagram without programming.

The third requirement will provide the ability to enhance the system capability according to the new requirement of users and to extend to new DSP areas.

Meeting the requirements above will allow users complete control over all aspects of the signal modeling process, such as algorithm selections, their sequence and internal parameters for each algorithm.

To meet the requirements mentioned above, four components were designed and implemented specifically, algorithm library, signal processing library, signal processing control tool and signal processing configuration tool.

The procedure of the tools was to designed to work in this way: First, users use the signal processing configuration tool to specify the sequence of algorithms and their configuration using a block diagram and save them to a file called a recipe file. Second, the signal processing control tool takes the speech data file and recipe file as input, parses the recipe file using functions provided by signal processing library to get the necessary information for each algorithm, applies the corresponding algorithm functions of each algorithm to processing input speech data by calling the correct method in the algorithm library.

All these four components involved are introduced individually in the following three sub-sections.

## 2.1 Algorithm library

The algorithm library is the lowest level component of the four components of the signal processing tools. It is a collection of algorithms implemented as C++ classes. There are two types of algorithms in this library. One includes basic DSP algorithms and the other includes support algorithms for high level manipulate data flow through block diagrams.

Basic DSP algorithm library includes the most commonly used algorithms, such as windows, filter, filter bank and energy. These algorithms are designed to provide general-purpose functionalities and users can use these fully tested algorithms in their own software or as a tool to learn basic DSP course. The algorithms which have been implemented to date in this category include: energy, filter, filterBank, window, cepstrum, fourier transform, spectrum, correlation, covariance, prediction, reflection, log area ratio, calculus. All those basic algorithms are the most widely used algorithms and provide basic modules for building complex front ends such as mel cepstra, perceptual linear prediction, filterbank amplitudes and delta features [2].

Support algorithms are mainly designed to be used in ISIP environment and help to increase the signal processing

ability in the tools. They are put into algorithm library mainly because they work in a similar way with basic algorithms such as processing data stream.

Support algorithms provide primitive debugging tools and the ability to manipulate feature streams. The algorithms in this category which have been implemented include:

Constant: allows a mechanism for applying global constants, such as the mean value of a signal, to a signal. This class is used extensively to implement algorithms requiring multi-pass processing.

Math: provides an ability to form weighted linear combinations of functions of feature vectors. This class is designed to provide maximum flexibility by supporting a mini-scripting language for functional analysis. It gives the front end a Matlab-like capability.

Statistics: used to compute means, variances, min, max, and other global measures of the inputs. This class is used to implement concepts such a mean normalization and variance-weighting. Since this class accumulates global values of its inputs, its interface is a little more complicated. Most statistical computations are inherently non-real-time, and require at least one complete pass over the data.

Connection: allows feature streams to be merged.

DisplayData: can be inserted anywhere in a signal flow block diagram to display feature values to the console.

Mask: allows individual features, or groups of features, to be selected from a stream.

Output: can be embedded in a recipe to output data to a file.

Generator: can be used to produce various types of signal.

CoefficientLabel: allows coefficient type of feature streams to be renamed.

AlgorithmContainer: used to hide details of specific components from the recipe processing software.

Each algorithm mentioned above provides multiple implementation options, such as in window class there are rectangular, blackman, bartlett, dolph\_chebyshev, gaussian, hamming, hanning, kaiser, lifter and custom window options. Users can choose any one. At the same time, such structures of implementation options were also designed to expand easily and users can add new implementation for each algorithm.

## 2.2 Signal processing configuration tool

A Java GUI tool called signal processing configuration tool as shown in Figure 1 was developed to provide users a block diagram approach to designing acoustic front ends. The Java language was used to allow the tool to run across a wide range of platforms (including Microsoft Windows), and to

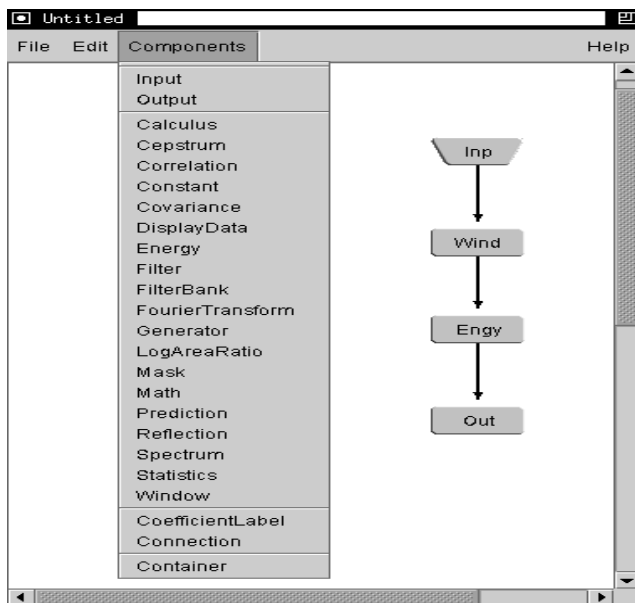


Figure 1: Signal processing configuration tool

give the tool an industry-standard look and feel.

The signal processing configuration tool was designed in such a way that users can draw block diagrams and connect each block using directed arc and configure each block in the diagram and save all the block diagrams into a file called recipe file. Each block represents one algorithm which the user can specify to process data; each arc represents a data flow from one algorithm to another algorithm. The format of a recipe file can be recognized and used by the signal processing control tool. The signal processing configuration tool also organizes the element in the components menu using a resource file. All algorithms appearing in the component menu (as shown in Figure 1) of signal processing configuration tool come from the resource file. This feature makes any new algorithm easy to plug in to the signal processing configuration tool by simply adding the description of the new algorithm into the resource file according to some predefined format. Users do not need to modify the source code of the signal processing control tool itself.

### 2.3 Signal processing library and control tool

The signal processing library was designed to manage the signal process. It is a collection of special designed components implemented as C++ classes to build a bridge between block diagram and computation algorithms. Its functions include: parsing the recipe file specified by users through the signal processing configuration tool; keeping synchronization for different paths along the recipe file; preparing input/output data buffer for each algorithm, especially when the multiple frames data are needed such as windows, calculus algorithm; scheduling the sequences of

required signal processing operations and processing data through data flow defined by recipe file; managing conversational data. Signal processing control tool provides some basic functionalities such as commandline parsing, multiple recipe files and multiple speech source files managing in one run. In this procedure, the work of signal processing library is hidden to most users.

All algorithms used in the recipe file are wrapped into component classes. The components can be processed uniformly. Such uniform structures and processing methods will make sure the extensible for new algorithm and the new algorithm can be plugged in easily. The base code of the system does not need to modify.

### 3. EXPERIMENTAL RESULTS

The tests for the tools include two aspects: one is to show how easy it is to construct block diagrams and save to a recipe file. The other is to verify the correctness of the computation results using the recipe file. We have successfully built several complex front ends with this tool, including an industry standard front end based on Mel-frequency cepstrum coefficients (MFCCs) [3]. The signal processing control tool supports multi-pass processing, which allows non-real-time research ideas involving complex normalization and adaptation schemes to be easily implemented. The block diagrams as shown in Figure 2 are constructed for MFCC with cepstral mean subtraction and energy normalization after getting cepstral mean and maximum energy by using another recipe file. To construct such block diagram is very easy. First, the user simply select algorithms from submenu of components menu and put them on the main panel. Then he connect them using the submenu functions of edit menu. Third, the user right-click the mouse and configure each algorithm according to the predefined specification. Last step is to save these block diagram to a recipe file. That's all need to do to create a recipe file. The user is ready to use signal speech control tool and the recipe file to process signal data.

The verification of computation is also perform extensively to make sure the correctness of the algorithms.

For each single algorithm implemented for signal processing it was fully tested using Matlab or manually to make sure it works correctly.

The feature vector results by using the recipe files are the same as the those from output from prototype system [4] of ISIP using the same specification. The prototype system have been used to many application successfully including Switchboard (SWB) [5], CALL Home [5], Resource Management (RM) [6] and Wall Street Journal (WSJ) [7].

The signal processing tools also combined with the decoder of ISIP production system. The TIDIGITS experiments were

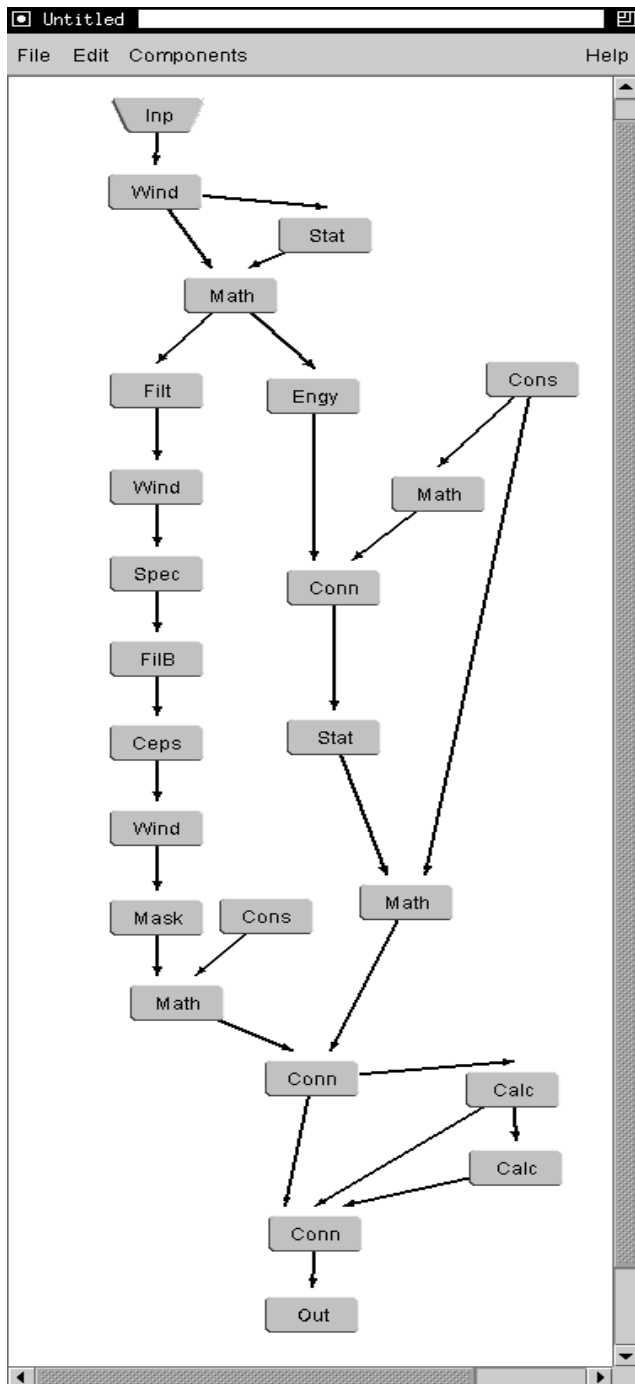


Figure 2: MFCC front end block diagram performed and got 100% correct result.

#### 4. PROGRAMMING INTERFACES

All the libraries presented above were developed to fit the requirements of the ISIP foundation classes (IFCs), which are a set of C++ classes organized as libraries in a hierarchical structure. Algorithm and signal processing libraries are two important components for IFCs. IFCs are

targeted for the needs of rapid prototyping and lightweight programming without sacrificing efficiency. Some key features include:

- unicode support for multilingual applications;
- math classes that provide basic linear algebra and efficient matrix manipulations;
- memory management and tracking;
- system and I/O libraries that abstract users from details of the operating systems.

The IFCs environment provides support for users to develop new approaches without rewriting common functions. The software interfaces are carefully designed to be generic and extensible.

The signal processing library is also designed in such a way that both the signal processing (front end) and recognizer share the exact same code base and have the same features.

#### 5. CONCLUSIONS

In this paper, we have presented an introduction to the signal processing tools which provide general-purpose DSP functions and serve critical roles in developing state-of-the-art public domain speech recognition systems. The motivation for the development of such an environment is to provide the community with a toolkit that can accelerate the process of developing new ideas and applications. The signal processing tools provide users powerful visual tools describing algorithms and rapid prototyping any new ideas without any programming. It also fits to the ISIP recognizer well and provide simple front end for recognizer.

#### 6. ACKNOWLEDGEMENTS

The original design of signal processing tools grew out of a series of discussions between R. Duncan and J. Hamaker. Significant improvements to the signal processing configuration tool have been made by J. Wang. There are a number of students and staff who have made significant contributions to the design and implementation of this software. We are deeply indebted to them for their hard work to make this system a reality.

#### 7. REFERENCES

- [1] K. Huang and J. Picone, "Internet-Accessible Speech Recognition Technology," presented at the IEEE Midwest Symposium on Circuits and Systems, Tulsa, Oklahoma, USA, August 2002.
- [2] V. Mantha, R. Duncan, Y. Wu, J. Zhao, A. Ganapathiraju and J. Picone, "Implementation and Analysis of Speech Recognition Front-Ends," Proceedings of the IEEE Southeastcon, pp. 32-35, Lexington, Kentucky, USA, March 1999.
- [3] N. Parihar and J. Picone, "DSR Front End LVCSR Evaluation - Baseline Recognition System Description," Aurora Working Group, European Telecommunications Standards Institute, July 21, 2001.

- [4] N. Deshmukh, A. Ganapathiraju and J. Picone, "Hierarchical Search for Large Vocabulary Conversational Speech Recognition," *IEEE Signal Processing Magazine*, vol. 16, no. 5, pp. 84-107, September 1999.
- [5] R. Sundaram, J. Hamaker, and J. Picone, "TWISTER: The ISIP 2001 Conversational Speech Evaluation System," presented at the Speech Transcription Workshop, Linthicum Heights, Maryland, USA, May 2001.
- [6] F. Zheng and J. Picone, "Robust Low Perplexity Voice Interfaces," <http://www.isip.msstate.edu/publications/reports/index.html>, MITRE Corporation, May 15, 2001.
- [7] S. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllable Word Recognition in Continuously Spoken Sentences," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357-366, Aug. 1980.