2006-472: A MATLAB TOOL FOR SPEECH PROCESSING, ANALYSIS AND RECOGNITION: SAR-LAB

Veton Kepuska, Florida Tech

Kepuska has joined FIT in 2003 after past 12 years of R&D experience in high-tech industry in Boston area in developing speech recognition technologies. Presented work is partially the result of the belief that cutting edge research can only be conducted with appropriate supporting software tools. In order to bring that cutting edge research to undergraduate level, the software tools have to be not only easy to use but also intuitive. Thus, presented SAR-LAB software was designed and developed with a clear goal in mind to evolve into a standard educational as well as research tool in the area of speech processing, analysis and recognition.

Mihir Patal, Florida Tech

As un undergraduate student Mihir was part of the NSF funded team develping MATLAB tool for Speech Processing, Analysis and Recognition: SAR-LAB. Mihir served a crucial role in design and execution phase of the project.

Nicholas Rogers, Florida Tech

Was an undergraduate student when involved in partialy fundened research in Machine Learning. As part of this research, MATLAB based research tool: SAR-LAB emerged. Nicholas played a crucial role in development of such a tool.

A MATLAB Tool for Speech Processing, Analysis and Recognition: SAR-LAB

Abstract

Presented work is related to research performed in developing a "smart-room." A smart-room can sense all voice activity within the room and pinpoint the source of the audio signal (speaker). The purpose of this audio sensing is two-fold: to monitor for key words, sentences, or phrases that are flagged as relevant by the monitoring entity as well as separation of all acoustic sources from each other (e.g., background noise from the speakers voice) Crucial requirement in successfully creating such a smart-room is the accurate (in terms of recognition performance), efficient (CPU and memory), and consistent recognition of speech (must work equally well for all speakers; i.e., speaker independent, as well as all acoustic environments). To achieve this goal it becomes necessary to develop tools that enable for advanced research in the area of speech processing, analysis and recognition, specifically in this case wake-up-word[i] [WUW] recognition. In developing such a system numerous tests of various system models are necessary. Modules ranging from audio signal processing functions and feature extraction, voice activity detection, pattern classification, scoring algorithms, etc., must be combined in order to perform speech recognition. Thus, a major hurdle in this area of research is the analysis, testing, verification, and integration of the individual functions required for speech recognition. To address the analysis and testing issue an appropriate software tool is developed using MATLAB environment that enabled unified framework for tracking the performance of all necessary functions of WUW recognition system. This framework can also be used for testing algorithms and other software components performing speech analysis and recognition tasks. In addition to integrating all of the various components, testing environment can produce additional analysis data all appropriately presented as graphs, charts or images (e.g., spectrogram) that are useful when analyzing and/or troubleshooting such components that are under research. This testing environment has proven to be very useful in aiding research in development of "wake-up word" recognition technology. This tool thus has made research process much more efficient, accurate, and productive.

Introduction

The primary objective of presented work was to develop a speech recognition engine - analysis and testing environment in MATLAB. The problem encountered when working with speech recognition projects is the fact that the processed data comes in the form of a large collection of vectors (e.g., matrix) that typically represent energies of a speech sounds at various frequency bands [1]. Developed testing utility is extremely useful because it provides visual representation of various complex parameters represented as patters, vectors or scalars extracted from time-dependent speech signal. In addition, there are various specific features, original or derived, that are traditionally difficult to analyze due to interdependency and time dependency. It is envisioned that the testing utility will greatly benefit future speech application developers due to its versatility and ease of extensibility. Other future uses include possible integration of the tool into newer versions of MATLAB.

To best describe the tool it is necessary to understand the architecture of typical speech analysis and recognition system. Thus the paper is organized as follows; first section depicts audio front-end module. Back-end of the system that contains speech recognition and scoring module of the system is described in section 2. Section 3 covers the details of the SAR-LAB; MATLAB tool kit. In section 4 development details of the implementation of the tool are presented. Conclusions derived from the experience of developing and using the tool is presented in section 5. Some possible enhancements that are planned in the near future are outlined in section 6.

1. Audio Front-End

WUW recognition system follows the generic functions depicted in the Figure 1-1. Speech signal captured by the microphone is converted into an electrical signal that is digitized prior to being processed by the WUW recognition system. The system also can read digitized raw waveform stored in a file. In either case raw waveform samples are converted into feature vectors by the front-end with a rate of 100 feature vectors per second – defining the frame rate of the system. Those feature vectors are used by the Voice Activity Detector (VAD) to classify each frame (i.e., feature vector) as containing speech or no-speech defining the VAD state. The state of the VAD is useful to reduce computational load of the recognition engine contained in the back-end. Back-end reports recognition score for each token (e.g., word) matched against a WUW model.

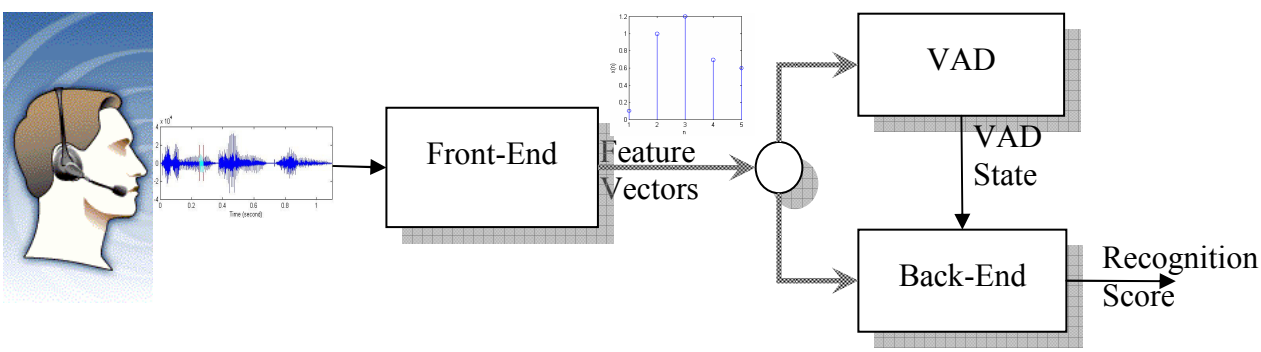

Figure 1-1 High-level representation of a WUW Speech Recognizer.

1.1 Audio Acquisition

Input to the system can be done via a microphone (live-input) or through a pre-digitized sound file. In either case the resulting input to the feature extraction unit, depicted in the Figure 1-1 as Front-End, is digital sound.

1.2 Feature Extraction of the Front-End

Feature Extraction is accomplished using standard algorithm for Mel Scale Frequency Cepstral Coefficients (MFCC). Features are used for recognition only when the VAD state is on. The result of feature extraction is a small number of coefficients that are passed onto a pattern

recognition algorithm. The feature extraction module generates a vector of these coefficients at regular intervals, and these coefficients represent the relevant properties of the audio signal at that point.

To perform any type of speech recognition, one of the first steps is to extract certain information about the speech signal in order to compare it to a model of some sort to see if what was said matches with a model representing basic phone (i.e., mono-phone, bi-phone, tri-phone or higher order context phone) certain word or even the whole phrase. There are number of ways in which to perform this feature extraction task [1], and the one used in the presented system is known as Mel Frequency scale Cepstral Coefficients (MFCC) [2]. This component is referred to as "front-end" of the whole system. The input to the front-end is a digitized speech signal and the output is a series of vectors that are representative of the signal. One may ask why such elaborate schemes to extract the frequency content of a particular sound are developed, since frequency content is merely given by the Fourier Transform of a certain signal. However, it is difficult to accurately compare two words in a meaningful way, or two sets of words when only looking at the pure frequency content of the two sounds in particular when considering that human auditory system is not equally sensitive to all frequency ranges [1]. All words have unique combination of sounds. However, the same word spoken by different people, or even the same person, can and will have different frequency distribution of the energies due to variation of resonances in the vocal tract caused by different physical characteristics of each person or different position of articulators of the same person at different times. Furthermore, timing of the uttered word is never the same even for the same speaker. With all the variability in a particular uttered word, it is not possible to perform a straight match between two word patters (i.e., test word pattern may not have the same number of vectors composing it compared to the model). In addition, the factors introduced by a noisy environment (all real-life environments introduce some amount of noise) change the frequency content of the acoustic patterns, thus altering the word represented by the pattern of feature vectors as seen by the computer as compared to its corresponding stored model. Those issues are addressed by the front-end to normalize as much as possible the pattern of a word by removing effects of the additive as well as convolutional quasi-stationary noise.

MFCC is a pre-processing algorithm that attempts to capture sound characteristics that are like human auditory system. The Cepstral Coefficients are determined by taking the natural logarithm of the magnitude of the Discrete Fourier Transform of the input signal, and then obtaining the Inverse Discrete Fourier Transform of the resulting sequence. In mathematical notation, the coefficients are represented by the following equation where $X(e^{j\omega})$ represents Fourier transform of a windowed segment of the signal:

$$c_x = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log\left|X\left(e^{j\omega}\right)\right| e^{j\omega n}\, d\omega \qquad \textbf{1-1}$$

Typically, 12 coefficients ($c_1, c_2, \ldots, c_{12}$) are combined with the signal energy to produce feature vectors. However, more features can be produced by taking the first and second derivatives of these values in order to represent changes in the signal. In presented-front end total of 42 (14*3 = 42) features are generated.

The diagram below (Figure 1-2) illustrates the process of extracting the feature vectors from a digitized speech signal.

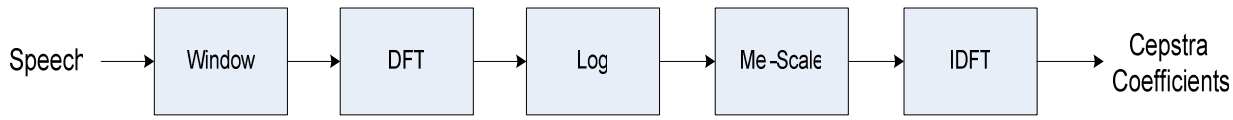

Figure 1-2 Signal processing by the front-end.

As can be seen from the above graph, by applying Mel-Scale, linear frequency of the signal content is transformed so that lower frequencies are more finely differentiated compared to higher frequencies. This mimics human hearing since human ear can differentiate better between two close sounds at lower frequencies (e.g., 200 Hz and 250 Hz) as compared to higher frequencies (e.g., 3200Hz and 3250Hz). Providing higher resolution of sounds in lower frequencies compared to higher frequencies make the recognition more accurate since it matches the relevance of each frequency band to human auditory processing. Furthermore, additional processing to this base procedure allow for reduction of convolutional noise (e.g., cepstral-mean normalization [3]).

1.3 Voice Activity Detector

The purpose of the voice activity detector (VAD) is to reliably detect the presence or absence of speech. This tells the front-end application, and thus correspondingly the backend, when and when not to process speech. The way in which this is typically done is by measuring the signal energy at any given moment. When the signal energy is very low, it suggests that no word is being spoken. If the signal energy spikes and stays at a high level for a considerable period of time, a word is most likely being spoken. Therefore, the VAD searches for extreme changes in the signal energy, and if the signal energy stays high for a certain amount of time, the VAD will go back and mark the point at which the energy changed dramatically. The energy of a signal is given by the following equation.

$$E_x = \sum_{n=-\infty}^{\infty} |x(n)|^2 = \int_0^1 |X(\Omega)|^2 \, d\Omega \qquad \textbf{1-2}$$

In order to achieve a more accurate VAD discrimination, extracted features are also used and combined to generate a single VAD feature [4]. Appropriately combined features enable for simplified discrimination logic and more robust VAD.

2. Back-End and Dynamic Time Warping (DTW)

The simplest way to recognize a *delineated* word token is to compare it against a number of stored word templates and determine which model gives the "best match". This goal is complicated by a number of factors. First, different samples of a given word will have somewhat different durations. This problem can be eliminated by simply normalizing the templates and the unknown speech so that they all have an equal duration. However, another problem is that the rate of speech may not be constant throughout the utterance (e.g., word); in other words, the optimal alignment between a template (model) and the speech sample may be nonlinear. The Dynamic Time Warping (DTW) algorithm makes a single pass through a matrix of frame scores while computing locally optimized segments of the global alignment path [5].

The resulting alignment represents the best warp in time between the reference word and the test word. By keeping track of back-pointers, the full alignment path can be recovered by tracing backwards starting from the last point in the matching matrix (*X, Y*). An optimal alignment path is computed for each reference word template, and the one with the lowest cumulative score is considered to be the best match for the unknown speech sample.

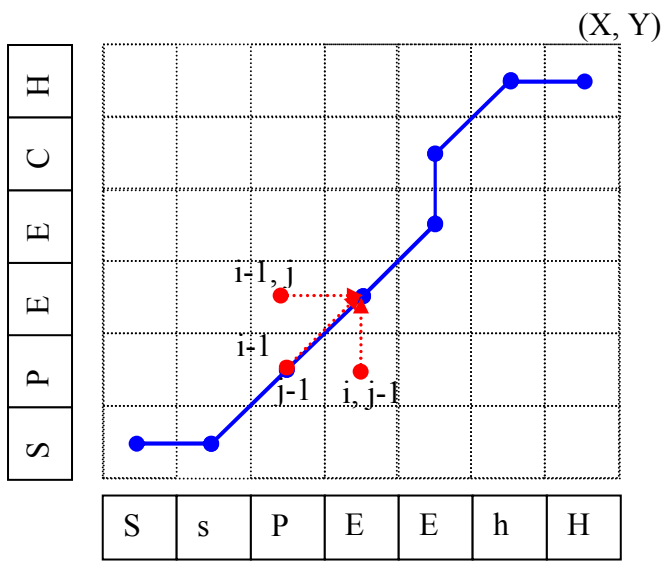

Figure 2-1 Simplified example of DTW matching of the model of the word "SPEECH" with the test token "SsPEEhH".

3. SAR-LAB

MATLAB was used to integrate all of functional components of interest of the system into a unified testing environment. MATLAB was chosen due to its ability to quickly implement complex mathematical and algorithmic functions, as well as its unique ability to visually display results through the use of image plots and other such graphs. Also, we were able to develop a GUI in MATLAB to use as the command and control interface for all of our test components.

At the core of our testing environment is the backend pattern matching algorithm. One of the goals of the presented testing environment was to research the effectiveness of the back-end algorithm; more specifically, an implementation of Dynamic Time Warping (DTW). The algorithm is used to perform speech recognition of a series of words against a speech model. Both a MATLAB script and C++ implementation of the DTW algorithm was developed. The C++ implementation was also combined with a MATLAB mex gateway function in order to compile a MATLAB-compatible binary executable. This allowed for faster execution of our DTW algorithm and thus the testing environment may be used more efficiently. The numeric result(s) of our DTW algorithm is fed-back into our testing environment and displayed in the GUI.

In addition to displaying the results of the backend algorithm, SAR-LAB produces a number of useful plots, graphs, and images. It includes images of test and model feature cepstral coefficients, audio signal and its spectrogram. Furthermore, SAR-LAB loads (or alternatively) computes the warped path representing optimal matching path and displays it as a graph on a two-dimensional matching grid off all possible matches. In addition, a number of auxiliary plots such as VAD features and warped energies can be generated on demand through GUI interface.

SAR-LAB reads-in the information generated by the C++ MFCC based speech recognizer system. This feature allows for analysis of the system performance and aids research. For example one useful information loaded (on demand) by the utility is the state of VAD module. Based on that information SAR-LAB determines which specific samples of an input audio waveform indicate a change in the VAD state. This information is used to iterate through each word/token within a test audio signal, resulting in the ability for the test environment to execute DTW matching algorithm on multiple words/tokens within a test signal vs. a model of a word. For each word within the test signal, a corresponding score, set of graphs, etc, are generated.

3.1 SAR-LAB GUI Description

To allow for easy interfacing with the testing environment, we developed a MATLAB GUI [6]. This GUI serves as a visually appealing, user friendly interface to our test environment. All plots and produced data are displayed through one, unified GUI (see Figure 3-1). Control features within the GUI include the ability to move between two adjacent words/tokens in the input screen, and optional buttons for auxiliary plots. The GUI also accepts filenames and directories for all test signals, models, etc. These configuration options may be automatically read in from a configuration text file.

Developed GUI allows for seamless integration of additional various testing functions as needed. This greatly supports the goal of performing analysis of speech signals, front-end processing as well as back-end recognition algorithms in a much more efficient and productive manner by placing emphasis on the logical representation and visualization of results, thus making analysis much easier, and testing less troublesome.

This seamless integration was achieved by utilizing MATLAB's GUI Development Environment (GUIDE), as well as the sharing of GUI-object meta-data generated by system's functional test

units. MATLAB's GUI allows for easy integration between various scripts, producing a more application-like feel for what would normally be a set of basic test harnesses.

Main SAR-LAB GUI is depicted in Figure 3.1. It contains various plots, spectrographic display and spectrographic like display of cepstral features. In addition it depicts the results of raw DTW matching of the baseline algorithm implemented in MATLAB code. In the lower left corner there fields depicting configuration file content that drives the GUI. This content can be edited offline and loaded or edited within the presented interface.

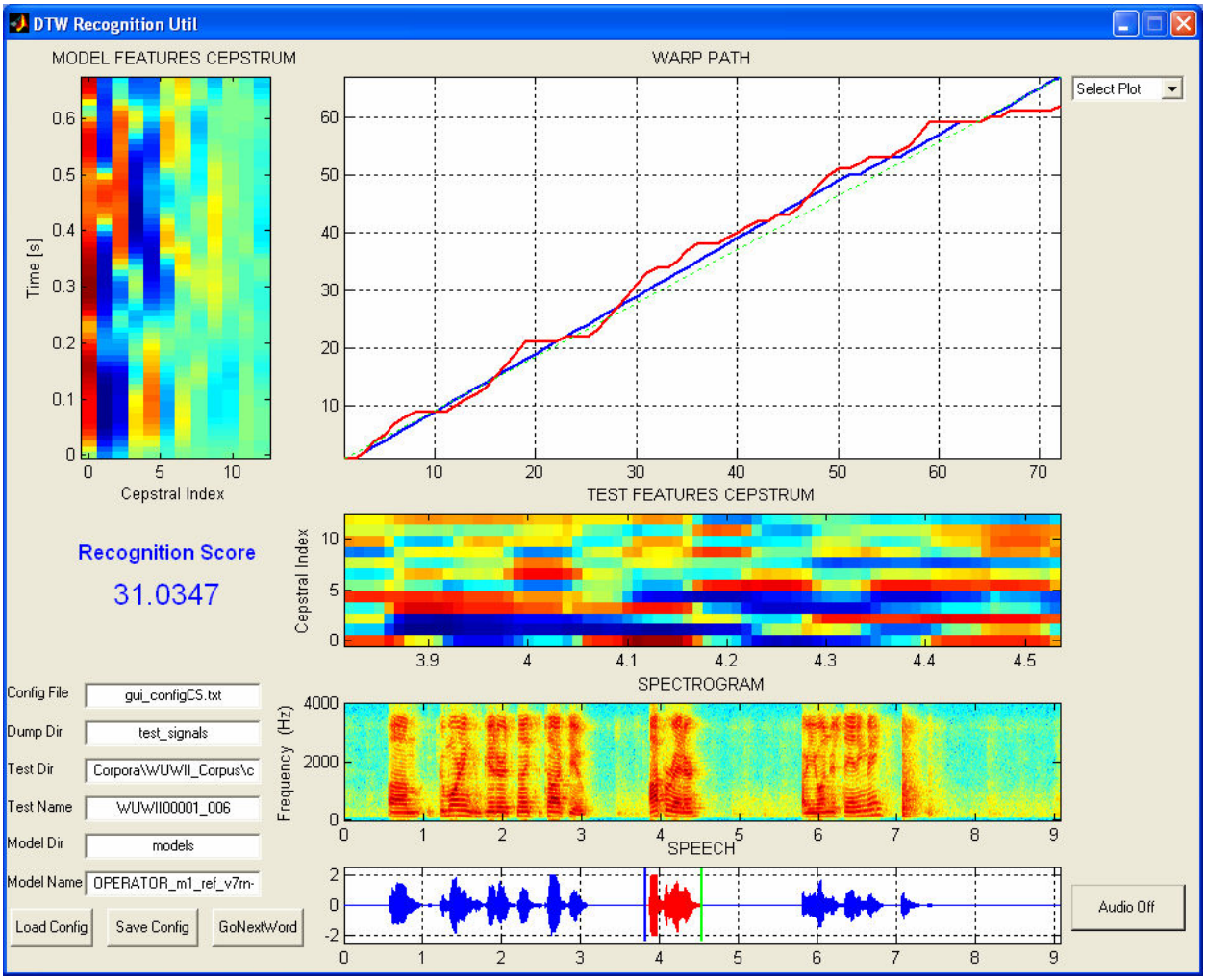

Figure 3-1. Main Interface of SAR-LAB. The WUW Recognition System selected token (delineated with the markers and red wave file) and its raw matching score of a WUW example.

At the top left corner of the window of the main SAR-LAB is the cepstral plot of the model against which the test signal is being compared. A cepstral plot is necessary here since the model does not have an associated waveform or spectrum – it has been obtained through training process by means of generalized averaging of appropriate train cases. Since it exists only as a collection of cepstral coefficients vectors, this is the only direct way to visually represent the model. Below the cepstral plot of the model is the score of the current warp. The current warp

refers to the part of the test signal that is being warped against the model at a particular instant in time. More on this will follow in the discussion pertaining to the test signal waveform. The score that is reported is the normalized score of the current warp. This represents the degree to which the test signal matches the model; the higher the score, the worse the match between the test and the model. The lowest score possible is 0, which indicates a perfect match between test and model. Realistically, this can only happen if the test signal features are the exact same as the model features. Below the score in the testing utility window are a group of input fields where the directories and filenames for the input arguments can be inputted. These directories and filenames are the location of the model and test feature files, as well as their respective names. The test feature file is generated by the feature extraction front-end. These directories and filenames can be specified by entering relative path names into the fields, or they can be read by a text file called gui_config.txt. This is the default configuration filename, and this can also be changed. The purpose of adding this function was to give the user a choice in the ways in which the input arguments can be loaded. Using a text file may be beneficial if the directories and filenames are fairly constant, thereby avoiding inputting the arguments in five separate fields.

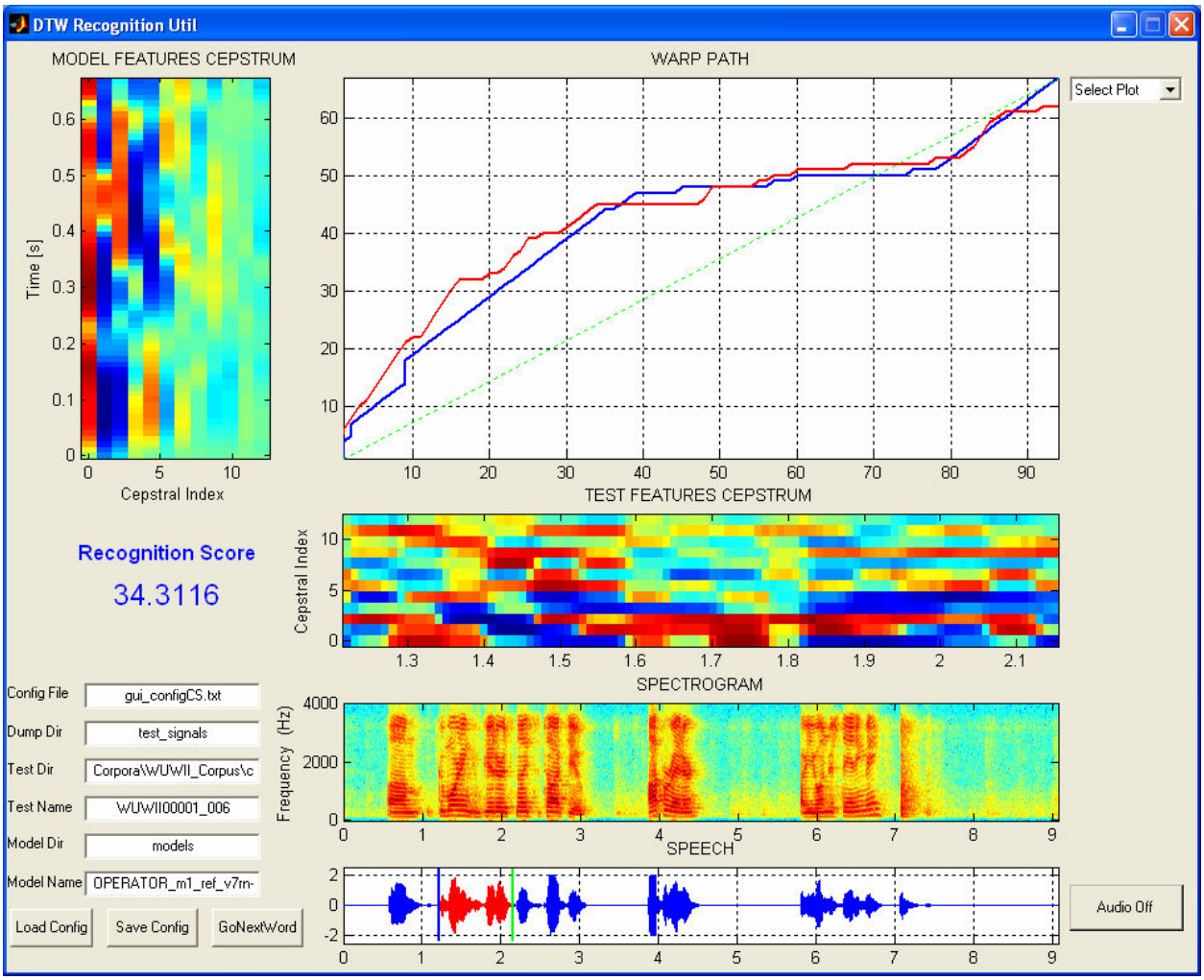

Figure 3-2. Next word/phrase token (delineated with the markers and red wave file) and its raw matching score that is not a WUW.

The plot that dominates the window is the DTW warp path. It is the result of the back-end recognition, and it gives a visual representation of the degree to which a test signal matches with a given model. The more diagonal the plot, the better the match between the two. A perfect diagonal line coincides with a perfect match and a raw score of 0. This is shown as a reference as the green line on the plot. The red and blue lines are the results of two implementations of the DTW algorithm: one in C++ and one as a MATLAB script. Below the plot of the warp path is the cepstrum of the test features. This is analogous to the cepstrum of the model, and is used as a comparison method to the model. Below this ceptrum plot is a plot of the test signal's spectrum. It is useful to see how this relates to the signal's representation in the time domain, which is shown below its spectrum. The waveform is a regular audio signal waveform that has been modified so that it shows the current matching word/token/phrase in red and the rest of the signal in blue. The portion in red is the portion of the signal that is currently being warped against the model. There are vertical markers on either side of this word as further indication that is the current matching word. To the immediate left of this plot is a button labeled, "Go/Next Word." This button initiates the start sequence of the DTW GUI and it also allows the user to move to the next word in the test signal. When the last word is reached, the program loops back to the first word when "Go/Next Word" is clicked. This allows the user to analyze the test an indefinite number of times.

3.2 SAR-LAB Additional Plots

The frame work of the SAR allows for easy addition of various plots. A number of such additional are implemented with a drop-down list implementation that can be depicted in the right upper-corner of the Figure 3-1 and 3-2.

This pull-down menu brings up separate figure windows, depending on the choice of plot. In each window, there are a number of unique plots which are all helpful in the analysis of a test/experiment. The first menu item labeled "VAD Feats + Energy" brings up the figure shown in Figure 3-3. The three plots shown in this window are the signal waveform, the VAD feature and the signal's energy as computed by the recognition system. This information is very helpful in the analysis of the VAD and its functionality. It can be clearly shown in this example that when the speech starts in the waveform, it correlates to a spike in the signal energy and consequently, a spike in the VAD features.

The second item in the pull-down menu on the far right of main window is labeled "Cepstrums," and it brings up a new figure as shown in Figure 3-4 and 3-5 of the corresponding to the two delineated segments of speech depicted in Figures 3-1 and 3-2 respectively. Each figure contains two sets of spectrographic like plots. First set from the top contains test and model cepstral features using their original time scale. Also note that the first two plots are the same cepstral plots as in the main window depicted in Figure 3-1 and 3-2 respectively. The second set of the features depicts optimally time-warped features to maximize their similarity. In other words both are compressed or stretched according to the degree of warping that takes place in DTW. As can be observed both test features and model features are warped appropriately by the DTW algorithm to achieve this goal (e.g., maximize similarity between patterns). In this case the test token does not contain the word against which model is being compared and thus the recognition is poor as indicated by a large matching score.

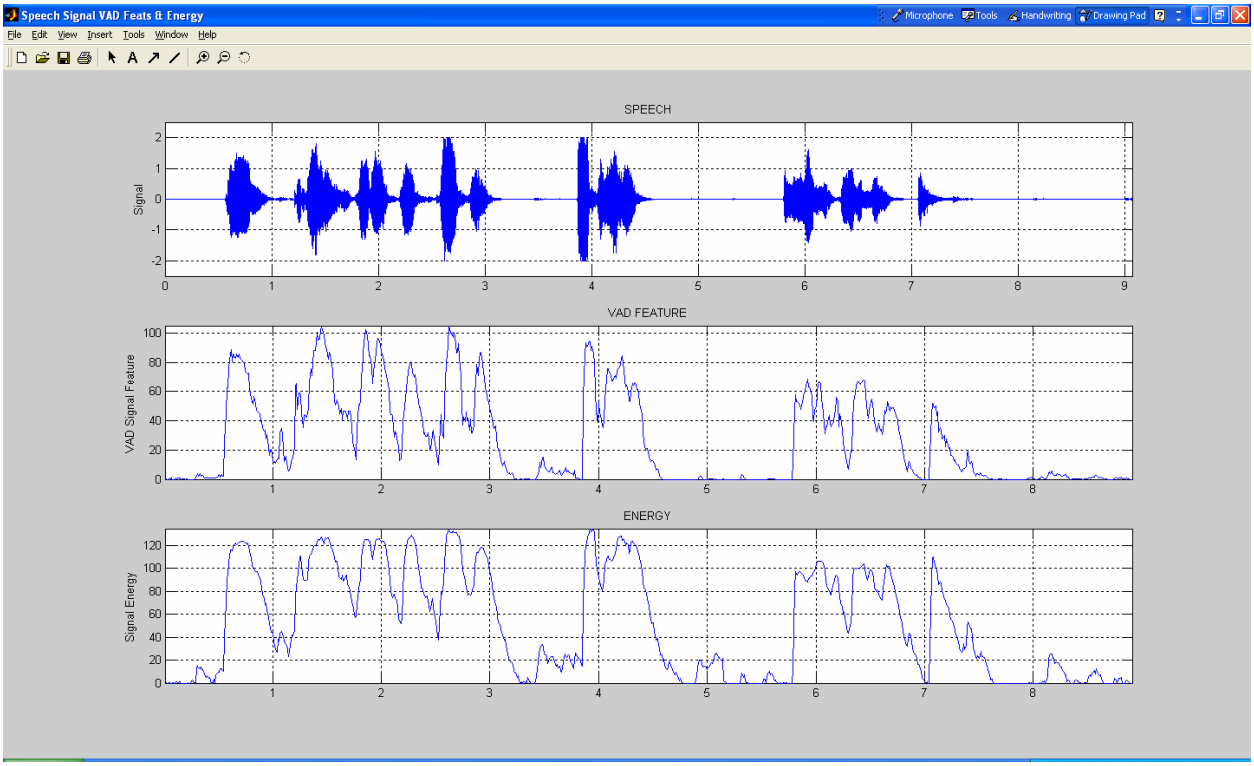

Figure 3-3. Supplemental analysis plots generated by selected from drop down list in the main GUI. First plot on the top depicts actual speech signal, followed by VAD feature and signal energy (i.e., c0).

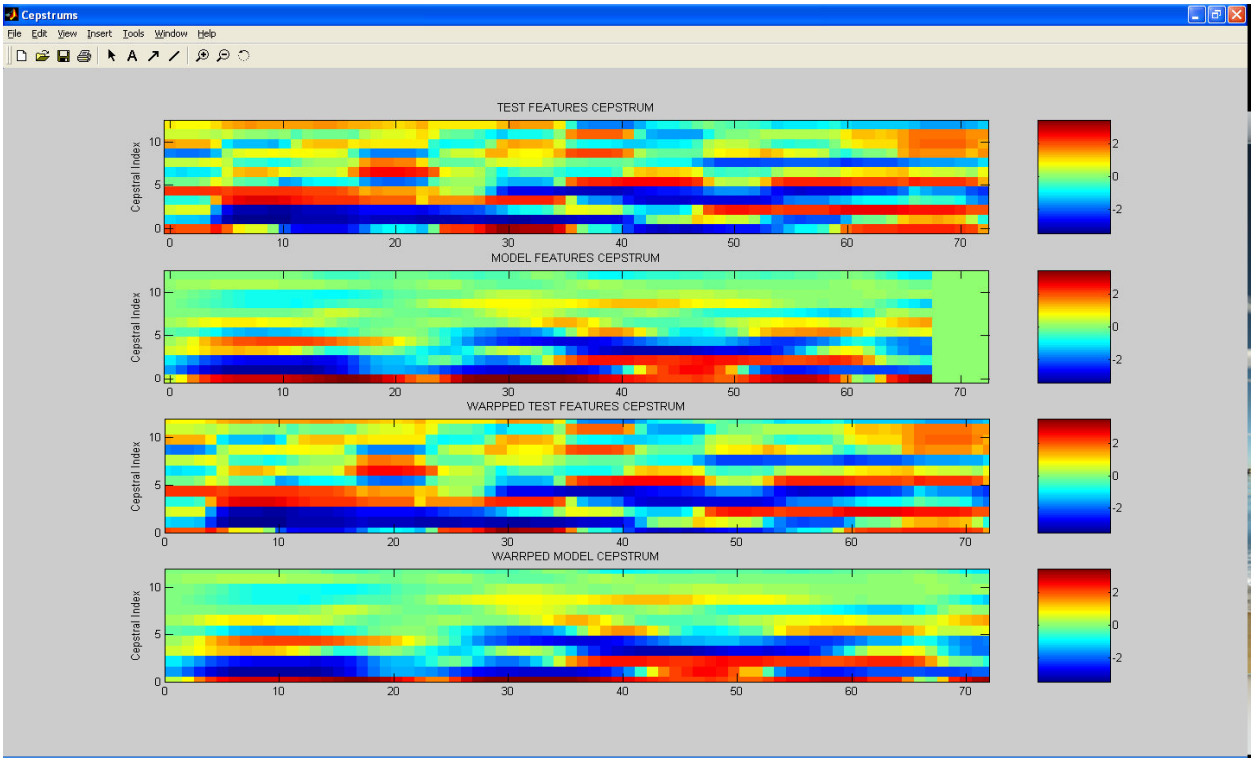

Figure 3-4. Original sequence of test and model features in the top and second plot depicting the segment presented in Figure 3-1. Warped test and model features by DTW algorithm to maximize similarity.

The third menu item in the pull-down menu, labeled "Energies" brings up a new figure, as shown in Figure 3-6 and 3-7. This figure window contains four energy plots. The first plot is the energy of the current word, or as the plot title calls it, the energy of the current test token. This is the energy of the part of the signal that is shown in red in the main window. The second plot is the energy of the model. This stays the same even after cycling through the other words, or tokens, as expected since the model is constant for a given experiment. The third and fourth plots show the first two plots, but with a warping factor being introduced. This warping factor is the same as that shown in Figure 3-1 and Figure 3-2 respectively, and it is calculated by the current DTW warp. In this plot it is more clearly presented the operation of DTW.

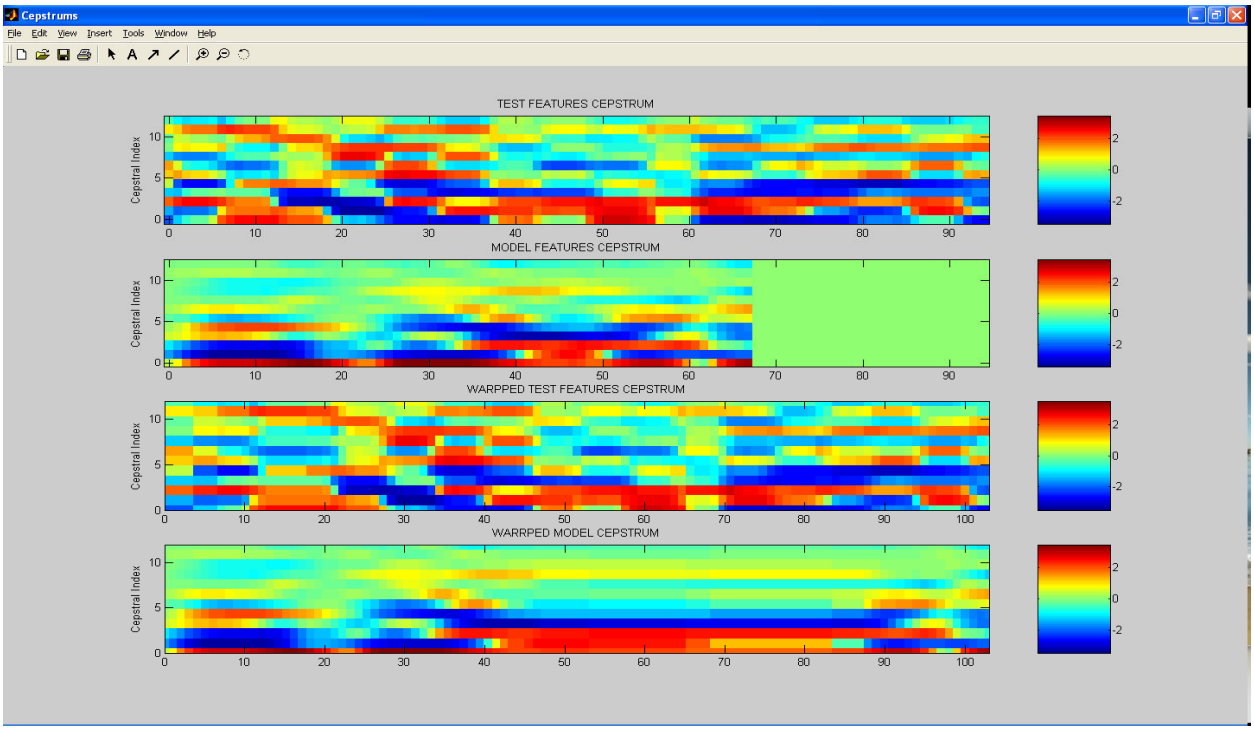

Figure 3-5 Original sequence of test and model features in the top and second plot depicting the segment presented in Figure 3-2. Warped test and model features by DTW algorithm to maximize similarity.

4. SAR-LAB Development

4.1 File Structure

The overall file structure of SAR-LAB is fairly simple. Within the directory, there are a number of MATLAB-associated m-files and a 'plots' folder, which contains all additional plots used in SAR-LAB. Of these m-files, there are two which control SAR-LAB: dtw_gui.m and main.m. dtw_gui.m is the associated m-file to the dtw_gui.fig file, which is the MATLAB GUI file. Dtw_gui.m contains all the callback routines to all GUI objects found in the .fig file. A number of these callback routines involve calling main.m, which is the file that computes recognition and all its associated plots.

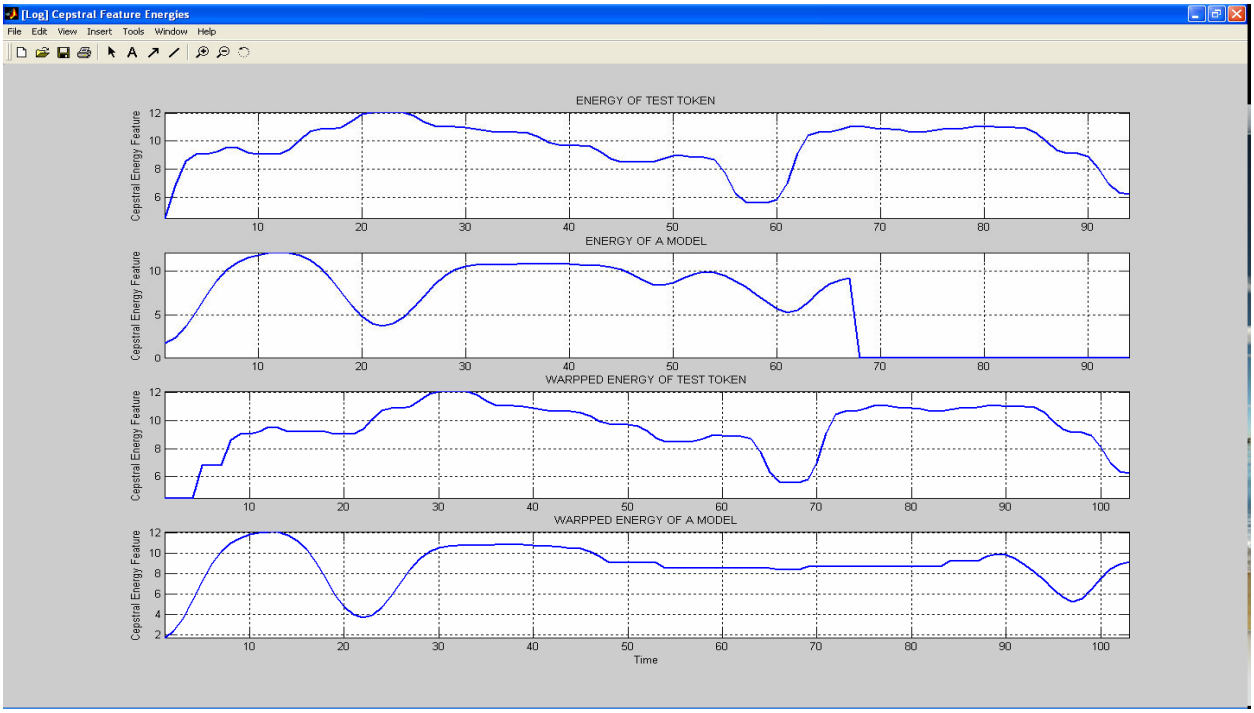

Figure 4-6. Original energy feature of the test token as well as energy feature of the model in the first two top plots. Bottom two plots depict warped test from the delineated segment depicted in Figure 3-1 and model feature.

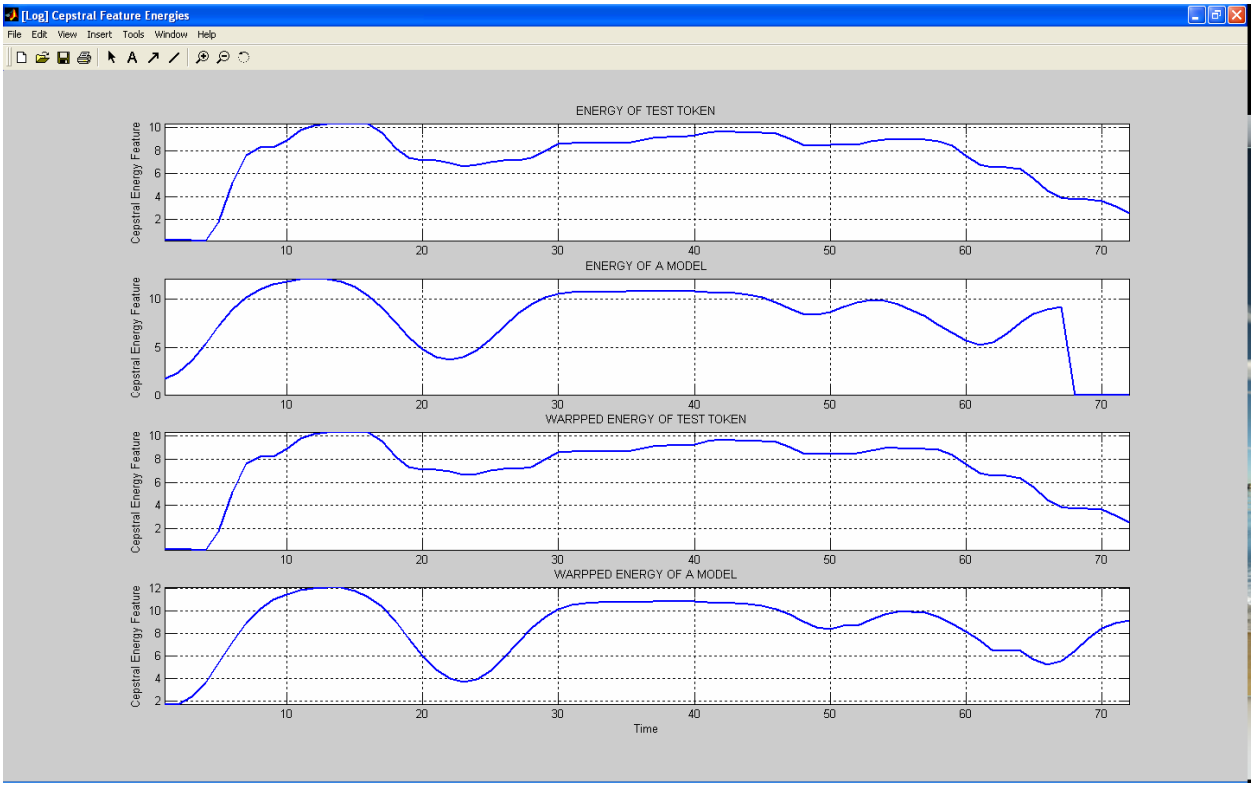

Figure 4-7. Original energy feature of the test token as well as energy feature of the model in the first two top plots. Bottom two plots depict warped test from the delineated segment depicted in Figure 3-2 and model feature.

This file contains all the code to call various functions necessary in the loading and recognition phases of the process. The most important of these functions are located in separate m-files to ensure a degree of modularity. All of the plots shown on the main screen of SAR-LAB are computed within main.m. The additional plots shown in the pull-down menu on the far right of the main screen are called by main.m, and their code is in separate m-files residing in the 'plots' directory within the main directory.

4.2 MATLAB GUIDE Description

To understand how to modify SAR-LAB, it is necessary to have a basic grasp and familiarity with MATLAB's GUIDE (GUI Development Environment). When a GUI is created using GUIDE, two files are generated upon saving: the .fig file of the actual GUI and the .m file associated with the .fig file. The .fig file basically shows the layout of all the GUI objects and the .m file contains all the callback routines for those GUI objects. When in GUIDE, the properties of any GUI object can be edited by double-clicking that object. This action brings up the property inspector. The 'tag' property of any GUI object is that object's identifier within the associated m-file, and consequently, it is one of the more important attributes of an object. The callback routine for an object is located in the associated m-file under a function whose name is that of the tag of that object. MATLAB's extensive help files contain plenty of information on GUIDE, including how to initialize, edit, and run a GUI. In addition to the help files, there are numerous books on how to use MATLAB which contain information on GUIDE [6].

4.3 Callback Functions File (dtw_gui.m)

The m-file associated with the MATLAB GUI (dtw_gui.m) contains all the callback functions for the buttons in SAR-LAB. These functions are all linked together with the 'handles' structure, which is a global structure that MATLAB uses to contain all the data for GUI objects. This structure can be used to share data within a file, or access GUI data, such as the string entered in a text field in the GUI. Since this structure is a global structure and it is already passed to all callback functions as an input argument by default, it is used extensively in our implementation of SAR-LAB. For example, once the file names and directories are obtained from the user via the text fields in SAR-LAB, the strings can be obtained from the handles structure, and this information is given another name and passed onto the main SAR-LAB file which performs all the recognition and plotting (main.m). In addition to sharing and accessing data from different GUI objects, the handles structure was used to set flags which would be tested in the recognition file. An example of such a flag is the word count flag, where SAR-LAB keeps track of the number of tokens in the input file and cycles through them each time the "Go/Next Word" button is pressed. This flag is initially set at -1 and upon every click of the button (including the first click), this value is incremented by two. It is incremented by two since every token is associated with *two* changes in the VAD state. This, along with other input arguments is used to call the function in the recognition file (main.m). Therefore, every time "Go/Next Word" button is clicked, or a plot from the pull-down menu is selected, the recognition file runs from the start with flags being set in a particular way according to the selection in the pull-down menu and the number of tokens in the input wave file.

4.4 Main SAR-LAB Recognition File (main.m)

This file contains all the code that performs the recognition between the input and test files. This file accepts as input arguments the file names of the input and test files, their directories, and the handles structure from the GUI file. The input and test data is then loaded and the variables for associated with that data are setup. Also, the current token of the input wave file is obtained. All this data is then fed to code that creates the plots for the input and test data. The data is also used by the code that computes and plots the DTW warp path. Finally, the file contains code that calls the functions for the additional plots in the pull-down menu. These plots are only activated by means of the flags setup previously in the callback file (dtw_gui.m).

4.5 Expandability

One of the unique aspects of SAR-LAB is its ability to be modified and tailored to the needs to any user. Since this GUI is based in MATLAB, expanding one of its current features only requires a rudimentary understanding of MATLAB programming and MATLAB GUIDE. The steps to add new plots are listed in point-form below.

Open the GUI file (currently named dtw_gui.fig) in MATLAB's GUIDE program. This is done by typing "guide" in MATLAB's command window and opening the existing dtw_gui.fig file. The GUI is then shown in MATLAB's GUIDE as can be seen in Figure 4-1. From here, any plot displayed in the utility can be modified. One of the more useful ways to expand the utility is to add additional plots to the pull-down menu on the right side of the GUI. This pull-down menu is an object in the GUI and its tag is "select_plot_popupmenu."

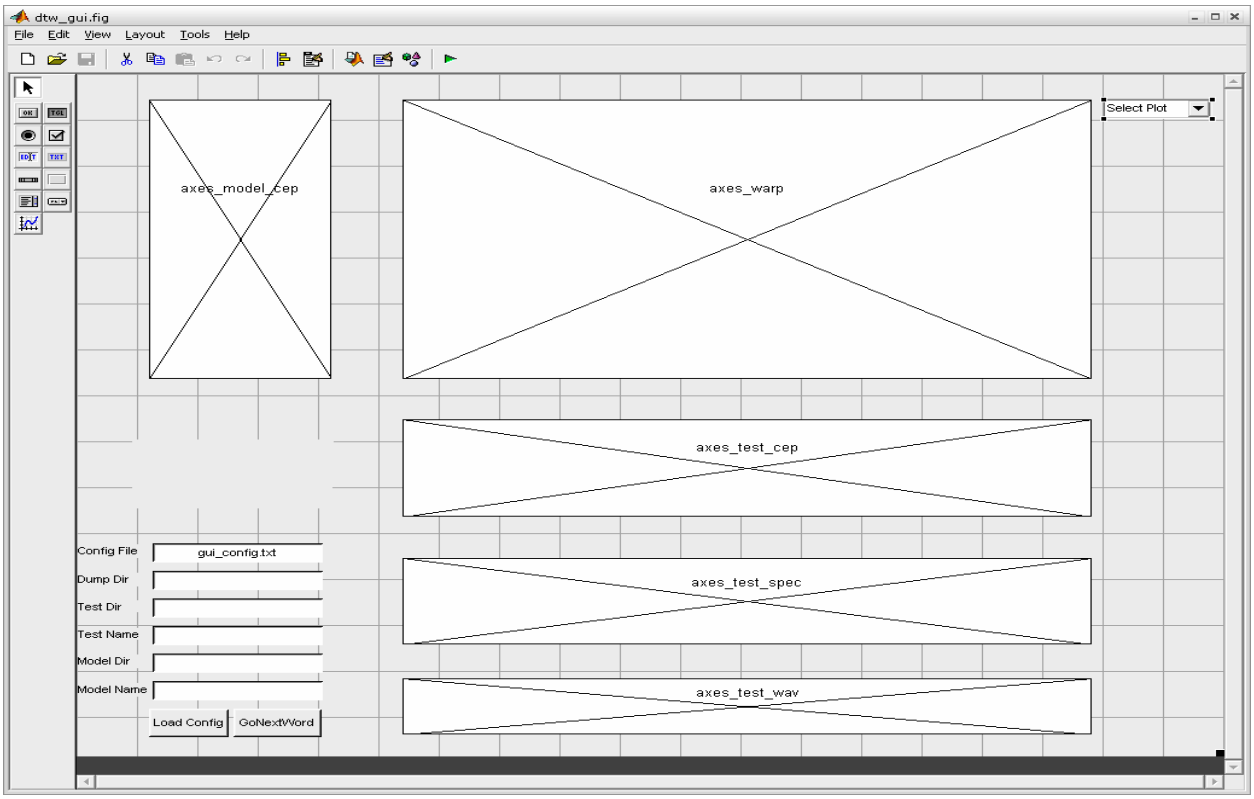

Figure 4-1. GUI Design using MATLAB GUIDE.

To edit the pull-down menu to add new plots, its string properties need to be edited. This is done by double-clicking the object in GUIDE to bring up the property inspector of that object. The property inspector for the pull-down menu is shown in Figure 4-2, along with the string properties window. The string properties window can be brought up be clicking on the icon to the immediate left of "Select Plot." The name of any new plot can be added to the list in the string window.

Next, a callback routine for the new plot must be added to the associated m-file of the dtw_gui.fig file (this m-file is named dtw_gui.m). This file contains a function named "select_plot_popupmenu_Callback," and under this function is a conditional switch statement that controls the plot selection from the pull-down menu in SAR-LAB. This switch statement contains cases which refer to the items contained in the pull-down menu. For example, "case 2" would refer to the "VAD Feats + Energy" plot. To add more plots to the pull-down menu, more cases would have to be added to the switch statement. A sample of a case is given in Figure 4-3 below.

The purpose of the switch statement is to specify cases in which certain flags can be toggled. For ease of implementation, the 'handles' structure was used to set flags in the main program of SAR-LAB (currently name main.m). The flag is set to 1, or high, if a certain case is selected, and this is then passed to the runMain function of dtw_gui.m. This function basically calls the script (main.m) that runs recognition and initiates the different plots. The flag information is passed to this script and it executes the code to generate certain plots depending on if a flag is 0 or 1 (ie. low of high). Once the main program is called, the flag that was set at the start of the case in the switch statement is then reset to 0.

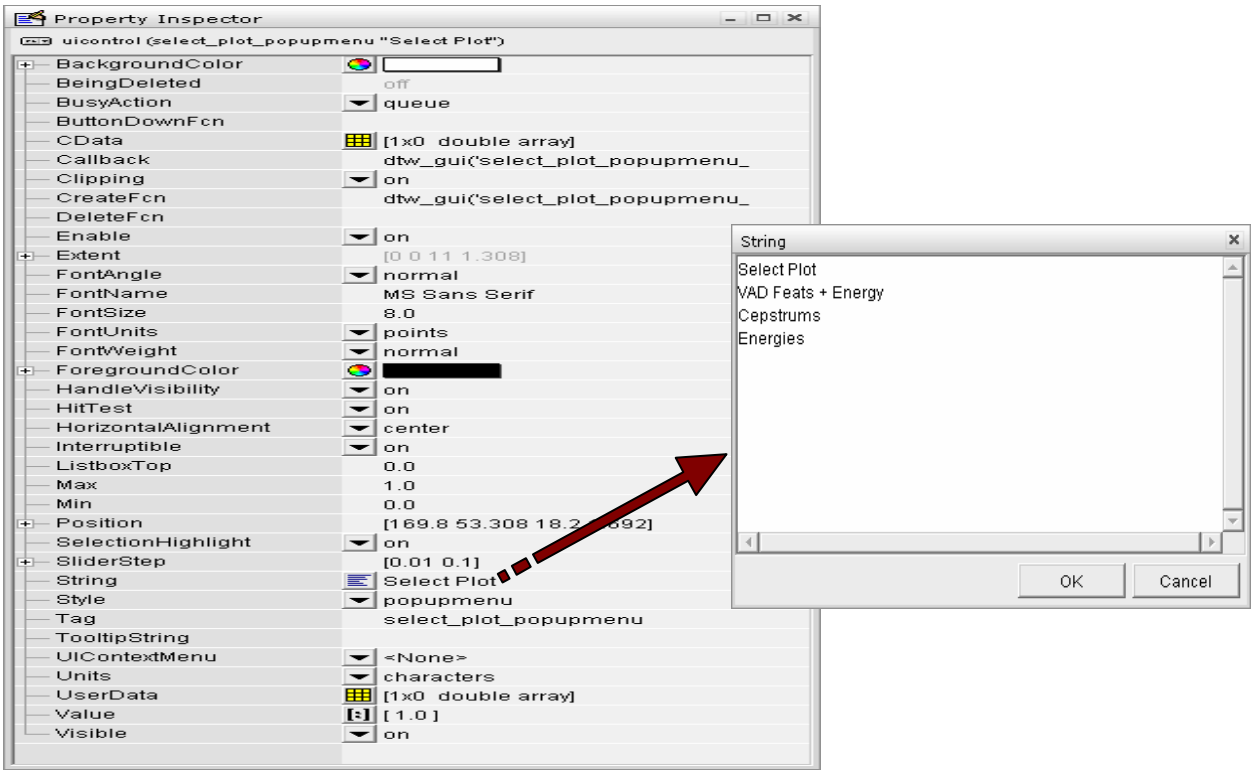

Figure 4-2 Property inspector pull-down menu

```
switch val
    case 1
    case 2

        handles.speech_plot_flag = 1;
        % save the changes to the structure
        guidata(hObject, handles);

        runMain(handles);

        handles.speech_plot_flag = 0;
        guidata(hObject, handles);
```

Figure 4-3 Sample code handling a case of additional plots in SAR-LAB

Once the necessary changes have been made to the switch statement in the dtw_gui.m file, some code would need to be added to the main.m file in order for it to respond to any changes in the flags. This code should be added to the bottom of the m-file, and a sample of this can be seen below in Figure 4-4.

```
% SPEECH SIGNAL, VAD FEATS, ENERGY AUXILIARY PLOT
if (handles.speech_plot_flag)
    fig = figure('NumberTitle', 'off', 'Name','Speech Signal VAD Feats & Energy');
    e_feature = TestFeatures(:,13);
    plot_speech(fig, wave, Fs, vad_feature, frame_rate, e_feature);
end

% CEPSTRUM AUXILIARY PLOT
if (handles.cepstrum_plot_flag)
    fig = figure('NumberTitle', 'off', 'Name','Cepstrums');
    plot_cepstrums(fig, InputFeatures, TemplateFeatures, warpped_test, warpped_model, WinputN, WtemplateN)
end
```

Figure 4-4 Example of code depicting plotting of additional figures in SAR-LAB set by switch statement in the previous figure.

As seen above, the code to call a plot is only executed if the flag for that plot is set to 1. The code to generate any plot residing in the pull-down menu should be located under the 'plots' directory within the program's directory. This code is user-defined and should be written to ensure compatibility with the rest of the SAR-LAB code (i.e. using input arguments that have already been calculated through the recognition in SAR-LAB).

Following these simple steps, the pull-down menu can be customized to almost any degree for numerous purposes. Furthermore, the plots are all called "on-demand," meaning a user would not have to sift through enormous masses of data produced by unnecessary plots for a particular task. Instead, single plots can be called when needed to display the relevant information. A user would also be able to customize the number of plots, as well as the types of plots that are shown. With a modular approach to code of the utility, we have also ensured that any user may be able to tailor the program to their needs with minimal effort.

5. Conclusion

The presented work describing SAR-LAB tool achieved two important goals:

- Provides facilities that enable process of research and development in area of speech analysis and recognition much easier, and
- It turn out that it is a great educational vehicle for demonstration of the key concepts important to speech analysis and recognition.

It is expected that this testing utility to be of great benefit to development of current and future speech applications by providing visual analysis of speech patterns. SAR-LAB was designed to be practical and extensible. In doing so invaluable knowledge of the concepts of general speech, speech analysis and recognition and MATLAB application development was gained.

In any kind of research, one of the most useful tools is the ability to quickly and efficiently analyze the results of a test or experiment during development cycle. This analysis usually includes the examination of several different plots and graphs, all of which visually represent data generated from an experiment. In speech recognition, examples of such plots and graphs are signal waveform, signal energy, spectral information of a signal, cepstral information, and warping path. All these plots give valuable information to researchers since certain anomalies can only be detected in certain graphs when appropriately presented. Therefore, the ability to generate all these plots, as a diagnostic tool, in an efficient manner can be a great help to anyone performing any kind of research in speech analysis and recognition. Furthermore, the ability to swap out different modules of a speech recognition engine without having to redesign a testing platform is a very big asset. We have achieved this by first standardizing our interface of data. This essentially means that once an effective and robust testing utility is created, changing different parts of the engine can be done with a fair amount of ease if the data follows prescribed standardization. Thus, there is no need to design and build another test-bed only adding additional plots if and when necessary. Our testing utility performs such a function. It allows researchers in speech analysis and recognition to quickly analyze the results of a test without going through the trouble of recalibrating the whole utility. Furthermore, through the use of MATLAB's GUI functions, interactivity and event driven functionality have been inherited. This tool also clearly emphasis the educational aspect and not only the benefit of the tool for research.

6. Future work

A natural extension to this tool is to enable it to analyze Hidden Markov Models (HMM's) [7]. A major contribution would be in development of an intuitive way to display functioning of the HMM, similar to back-tracking path of DTW although with the addition of the HMM states. Further functional flexibility can be added or enhanced; for example generation on demand of a spectrographic slice from selected time instance of the speech, etc., would render the tool more flexible and useful. The tool performance can also be improved by not performing redundant computations or operations. Optimization of the code will speed up the tool and improve its usability for tasks that require large analysis data files. Recently this tool was found useful in analysis of signals generated by seismic sensors, ultimately aiding development of a recognition system capable of detecting events embedded in the signal.

7. Acknowledgement

This work was partially supported by the National Science Foundation CCLI-EMD grant #0341601. The MATLAB code and sample data files generated by WUW speech recognition system are available from www http://my.fit.edu/~vkepuska/EMD-MLR/SAR-LAB/.

8. References

[1] L.R. Rabiner, R.W. Schafer, "Digital Processing of Speech Signals", Prentice-Hall, pp396-461, 1978

[2] S.B. Davis & P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences", *IEEE Trans. on ASSP* **28**, pp357-366, 1980

[3] Liu F., Stern R., Huang X. and Acero A. "Efficient Cepstral Normalization for Robust Speech Recognition". Proceedings of ARPA Human Language Technology Workshop, March 1993.

[4] Veton Këpuska et al., "Voice Activity Detection " Patent No. 13323-011001 - 10/144,248, 5/14/02

[5] C. S. Myers and L. R. Rabiner, "A comparative study of several dynamic time-warping algorithms for connected word recognition", *The Bell System Technical Journal*, 60(7):1389-1409, September 1981.

[6] Duane Hanselman and Bruce Littlefield, "Mastering MATLAB 7", Prentice-Hall, 2004

[7] Rabiner, L. R. and Juang, B. H., "An introduction to Hidden Markov Models", IEEE ASSP Magazine, pp. 4-15, January 1986

---

[i] Wake-Up-Word [WUW] Recognition is defined as a Speech recognition task of:
Correctly responding to a specific word/phrase when spoken, and
Correctly rejecting all other words/phrases/sounds/noises as not being that WUW, while continuously monitoring all acoustic events.