Visualization of Optical Speech Prosody

GRADUATE PROJECT REPORT

Submitted to the Faculty of
the Department of Computing Sciences
Texas A&M University - Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Rafal Dopierala
Summer 2007

Committee Members

Dr. Scott A. King
Committee Chairperson

Dr. Hongyu Guo
Committee Member

Dr. David Thomas
Committee Member
Visualization of Optical Speech Prosody

GRADUATE PROJECT REPORT

Submitted to the Faculty of
the Department of Computing Sciences
Texas A&M University - Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Rafal Dopierala
Summer 2007

Committee Members

Dr. Scott A. King
Committee Chairperson

Dr. Hongyu Guo
Committee Member

Dr. David Thomas
Committee Member
ABSTRACT

Work is presented on a Visualization Tool that has been created to explore how prosody (syllable accent, volume, intonation, and rate) affects the surface of the face. Research, being conducted at Texas A&M University Corpus Christi, works to create realistic animated characters. The research uses muscle-based parameters to animate a facial model over time.

Data was collected from previous work, which supplied the muscle-based parameters needed for this project. The data was used to compare the non-linear and linear relationship between prosody, and the change in shape. The Visualization Tool switches from numbers that cannot be easily interpreted by humans, into visual images in the form of curves. In order for the comparison to occur, the parameter values need to be examined in visualized curves. These curves can be easily seen by the user, and allows the user to compare and contrast data more efficiently.

When the Visualization Tool was implemented, it encountered some obstacles. The tool was designed in MatLab; however, due to limitations, a special data structure had to be designed in order to meet the requirements. The design includes both scripts and functions, but when functions are used, they have to be treated in a special way. Because they use their own workspace, the variables are not visible to other functions. A special structure had to be created, in order to make a database available for each function.

The work completed serves the research that strives to create realistic animated characters. Utilizing the Visualization Tool to conduct curve analysis increases the possibility for future exploration and expansion.
# TABLE OF CONTENTS

ABSTRACT .................................................................................................................. II

TABLE OF CONTENTS .............................................................................................. III

LIST OF FIGURES ........................................................................................................ V

1. BACKGROUND AND RATIONALE ........................................................................... 1

1.1. BACKGROUND ..................................................................................................... 1

1.2. DEFINITION OF PHONEME .............................................................................. 6

1.3. PRIOR RESEARCH .............................................................................................. 6

2. VISUALIZATION TOOL ............................................................................................ 10

2.1. PROSODIC VISUALIZATION ............................................................................. 13

2.2. REQUIREMENTS FOR SYSTEM DESIGN ............................................................ 19

3. VISUALIZATION TOOL DESIGN ............................................................................. 23

3.1. IMPLEMENTATION OF PROSODIC VISUALIZATION ........................................... 23

3.2. IMPLEMENTATION OF REQUIREMENTS FOR SYSTEM DESIGN ......................... 27

3.3. PROJECT EQUIPMENT REQUIREMENTS ............................................................. 29

3.4. PROJECT INPUT REQUIREMENTS .................................................................... 29
4. EVALUATION AND RESULTS ................................................................. 32

5. CONCLUSION AND FUTURE WORK .................................................... 42

ACKNOWLEDGEMENTS ........................................................................... 44

BIBLIOGRAPHY AND REFERENCES ....................................................... 45

APPENDIX A – PROJECT REQUIREMENTS ............................................ 46

APPENDIX B – PROSODY CORPUS .......................................................... 48

APPENDIX C – PARAMETERS DESCRIPTION ........................................ 49

APPENDIX D – SOURCE CODE ............................................................... 50
LIST OF FIGURES

Figure 1.1 The Proposed Speech Synthesis System.................................................................3
Figure 1.2 A Facial Animation System General Schematic......................................................7
Figure 1.3: A Parameter Control Screen..................................................................................8
Figure 2.1: Proposed Prosody Visualization System..............................................................10
Figure 2.1 Multiple Curve Display Visualization.................................................................15
Figure 2.3 Multiple Curve Groups Display Visualization....................................................16
Figure 2.4 Curve Extensions..................................................................................................17
Figure 2.5 TriPhone ‘M EH1 R’..........................................................................................18
Figure 2.6 All Parameters Display for A01...........................................................................19
Figure 3.1 Sample Database File........................................................................................30
Figure 3.2 Sample Parameter File........................................................................................30
Figure 4.1 Curve Two Groups No Extension OW0 and OW1..................................................33
Figure 4.2 Curve Two Groups with Extension EH for A03 and B03.......................................34
Figure 4.3 Curve Two Groups with Extension B.....................................................................35
Figure 4.4 Curve Two Groups with Extension ER0, A04 and A07........................................36
Figure 4.5 Curve Two Groups with Extension EH1 word marry..........................................37
Figure 4.6 Two Curve Comparison from Different Files D03 And B03..................................38
Figure 4.7 Curve Comparison from Different Files, Different Syllables and Aligned..............39
Figure 4.8 Curve Comparison from Different Files, different syllable and not aligned..........40
Figure 4.9 All length of the data display files 01, A, B, C and D............................................41
1. BACKGROUND AND RATIONALE

One way to work towards eliminating the barriers that exist between humanity and technology is to create better and more effective system interactions. One project that strives to accomplish this goal is TalkingHead, a research project, that is being conducted at Texas A & M University Corpus Christi. TalkingHead implements real-time, advanced, human face animation.

The primary goal of this project was to contribute to TalkingHead, by examining optical prosody, which affects the visual signal. This is explored in order to improve the model of prosody.

The influence of linguistics on speech patterns can be determined by examining speech prosody. Speech prosody “refers to the suprasegmental features of natural speech such as rhythm and intonation. Native speakers use prosody to convey paralinguistic information such as emphasis, intention, attitude, and emotion” [Chen 2006]. Just as paralanguage directly influences verbal communication, prosody influences not only the message, but the physical features of the message as well. This prosodic visualization is referred to as optical prosody. Studying optical prosody is imperative in order to ensure that realistic facial features can be created in artificial human systems.

1.1. Background

When a person speaks, he or she is always speaking more than words. It is impossible to take all that makes a person “human” out of the language that they speak. Emotion, mood, tone, diction, rate, etc… all contribute to the spoken message. The phrase, “it’s written all over your face” is sometimes used to intimate that the message has been sent without words. Another
phrase is, “it is not what you said, but the way that you said it” that impacted the message. Giving these somewhat cliché phrases their due, there is much more to language than words. If you can’t take the “human” out of the language, then how do you put the “human” into artificial human systems? One way to create realistic facial features is to take into account the entire message, including paralanguage and speech prosody. By examining every element that makes human interaction possible, one can then begin the process of analysis and incorporate the information into a program.

Some research in the area of prosody involves researching the text to speech synthesizer while inputting prosodic features. Much of the outcomes have been highly successful. One example of such research is the implementation of generating prosody from emotion. Researchers Mori, Moriyanna, and Ozawa, have linked speech prosody to emotion and based the research on the assumption of that link [Mori 2006].

Figure 1.1 illustrates the incorporation of prosody into synthetic speech. This system was created in order to improve synthetic speech by adding features such as anger, surprise, disgust, sorrow, boredom, and joy. To incorporate this feature, the system is calculating eigenvectors with corresponding eigen values. This is used to calculate the prosody pattern of the training samples.
These calculations, when combined with regression coefficients and target emotions, “linearly transform the specified emotion vector into the prosody parameters” [Graf 2002]. After modifications, this figure can be used as a base to construct the visual affects of prosody. This aligns with the belief that prosody has a direct impact on the physical facial features and that it needs to be examined more closely.

When a person speaks their face does not remain still. There are always facial movements that are tied into the message. A person might furl their brow, bite their lip, or make some other exaggerated facial expression. In addition to the obvious movements, there are also slight movements that occur due to speech prosody. These movements occur all over the face [Mori 2006].
There are three approaches to resembling mouth movement, artistic, simplified synchronization and prosodic inclusion. 1) The artistic approach occurs when very skilled artists recreate the natural speech. This process is extremely expensive and happens in the movie industry under the watchful eye of graphic and animation artists. These highly skilled artists are difficult to find in every geographic location and are in high demand in their field. This lack of supply for the demand of creating realistic and natural graphics contributes greatly to the phenomenal cost of this type of project. Because the work is done through manual labor, it is also a painstaking process which takes time to produce and perfect. 2) The simplified synchronization approach occurs when simplified movements are made in synchronization. This method does not consider prosodic messages and is not scientific in nature. This method is used primarily for entertainment and fun and attracts children. This technique depends greatly on its audience and does not converse or communicate in a close setting. This animation typically entrances the audience in a plot, and in turn they do not pay close attention to the lack of some facial movements. This approach focuses on one way communication and does not intend to be international or promote the natural communication process. 3) The prosodic inclusion approach is a less researched area. The speech field researches this area in order to improve synthetic speech. To do so, prosody must be incorporated in order to include emotion, intonation and other paralinguistic factors. Most do not visualize the prosody itself which makes prosodic inclusion a cutting edge approach to solving more natural interaction. More possibilities are opened with the advancement in technology and people are normally experts with judging the quality of signals connected to the message; therefore, the need for prosodic inclusion in speech interaction has increased. This is to say that there is an expectation that truth be justified based on the physical features of the speech as well as the language itself.
Most of the changes in visualization have been implemented by the movie industry, which does not have to work in real-time. Today, there is a greater need for real-time interactions which even the most skilled artist is unable to provide instantaneously. As advancements in technology have increased, so has the anticipation of its users. Users have become more impatient when waiting for computer interaction to occur. New graphic cards, such as the NVIDIA GTX 8800, incorporate parallelism and advanced computational powers that allow for much faster and improved online graphics [NVidia 2007]. As the computational power of graphic cards increases, so will the possibilities for better and faster calculations and therefore, an increase in the user expectations. At every moment, there is a process of utilizing what is available now in order to meet the needs of the future. Watching a movie with outstanding graphics can be breathtaking, but watching this same move in five to ten years can become more like a joke because graphics advance and make the old ones look less authentic. One example of this is the movie Star Wars. In the original picture, Jabba the Hutt did not have all of the desired features because it was too difficult to achieve. In the redone version, technology has allowed artists to go back and add features that were left out.

The same is true with advancements in artificial human systems. “As talking heads look more and more like real, recorded humans, viewers become more critical of small deviations from what is considered natural” [Graf 2002]. The urgency to examine the fine details that contribute to facial movements is evident in the demand for higher quality, more natural, artificial human systems. Real-time, real life, interaction attempts to take the machine out and replace it with the illusion of interacting with a real person. TalkingHead is built on parameters which incorporate muscle movements into the face. Visual prosody includes all features such as eyebrows, cheeks, etc…, which are achievable. The most important part of integrating prosody
into the visual animation of the head is the movement of the mouth muscles. This area of concentration must have good visual speech movements which involve the tongue, the lips, and other parts of the mouth.

1.2. **Definition of Phoneme**

A phoneme is defined as “a speech sound that distinguishes one word from another, for example, the sounds ‘d’ and ‘t’ in the words ‘bid’ and ‘bit.’” [Encarta 2006], and it is perceived as the smallest phonetic unit of acoustic speech which depends on the phonetic track itself [Zoric 2005].

1.3. **Prior Research**

Although prior research has touched on speech prosody, it has done so in the area of hand and head gestures [Graf 2002]. Prosody in the proposed systems is used in order to improve synthesized speech. Basically, the emphasis has been placed on the obvious implications that speech prosody might have on a voice or gesture. One area in which research seems to be lacking is the affect of prosody on the visual features of the mouth muscles. There is ongoing research which involves a human computer interaction system called TalkingHead [King 2001].

1.3.1. **Talking Head Model**

TalkingHead animation model is a 3D animation which is based on a set of parameters and is controlling the formation of the system.
The schematic in Figure 1.2 shows the use of parameters to deform the original head based on the initial parameters. Parameters are used to achieve a new model shape which, combined with geometry, are rendered into an image. On the first step, the system takes the initial geometry and inputs it into the facial model. Then on each step of the calculation, the facial model uses preprocessed geometry to calculate an image [King 2001].

The muscle-based parameters which TalkingHead uses can be extracted to see how groups of phonemes affect the parameters themselves. Normally, target areas can be set up for every phoneme and the mouth can be deformed accordingly. This information was used in an earlier part of TalkingHead project conducted by Stan Leja and Dr. Scott King [Leja 2006]. This part of the project involved Dr. Scott King having his face marked to indicate specific facial movements while he was videotape speaking predefined sentences and words. The phonemes were able to be captured as motion data, which is for each marker; its 3D location is captured at 120 Hz. The sentences and words were parsed using Carnegie Melon phoneme definitions. The waveform signal was processed by Festival, which provided the basis of the timing information.
used to parse the captured sentences, by EMU labeler [Cassidy 1996]. The output of EMU was used to create a database for the proposed project.

1.3.2. Structure of Talking Head Lip Model

Figure 1.3 shows the graphical user interface (GUI) of TalkingHead. The sliders are connected to the controlling parameters [King 2000]. TalkingHead has many expansive areas of concentration; however, this project will concentrate only on the lip model and its parameters.

Figure 1.3 shows TalkingHead with the parameters that will be used in the project and these parameters are isolated in the lower left corner of the figure. These are the sliders that control the lip and mouth portion of TalkingHead. The parameter values range from -1 to 1 and
some are from 0 to 1 [King 2001]. This facial model is using a muscle-based parameterization [King 2005]. The values of TalkingHead parameters are extracted from the system and are used as a basis for comparison which makes it possible to search for the non-linear relationship between prosody, and the change in shape by looking at the parameter values as a curve. This project will provide a Visualization Tool which will help to expedite the comparison.
2. VISUALIZATION TOOL

This section provides a detailed description of the scope of the project and what it accomplishes. This proposal aids TalkingHead research by helping to determine how prosody affects the visual signal. This section is divided into two main sections: 1) Prosodic visualization and 2) System design. The areas that are discussed in this section are based on the requirements (see appendix A) and include: Displaying a single curve, displaying multiple curves, distinguishing groups of multiple curves, curve extensions, triphones, comprehensive parameter displays, system design, user friendly, query possibilities, input formatting, display versatility, and user interface and output. The visualization is written using MatLab.

![Block Diagram]

**Figure 2.1: Proposed Prosody Visualization System.**

The block diagram in Figure 2.1 is a visual representation of the project as a whole. The user interface is designed for accomplishing the task very fast and it takes into account the type of user. The system is designed for the command line user, which gives fast access to the
outcome. I had to choose between prompting the user for input and a command line function with input as parameters. The first option was a good choice and user friendly; however, it slowed down the overall process and therefore was discarded. I went with the second option, using a command line function with input as parameters, in order to expedite the output. This option did not compromise the user friendliness, because of the added help function. This help function is equipped to display a detailed manual for each function. Anytime the user needs to access the help information, they have to type on the MatLab command line “help” followed by the function name. In order to start using the system, the user has to make some preparations. The preparation phase includes, making the Visualization Tool folder accessible to the MatLab. This is done by setting up the path to the folder inside of the MatLab software. In order to accomplish this, after the user runs the MatLab software, they have to go to the tab file on the menu bar and then go to the set path. Next the user must press on the button, add with subfolders, browse to the folder and highlight it, then press okay. Next the user must press save on the set path window followed by close on the set path window. This set up has to be done only one time and does not have to be repeated on a private computer. This instruction set could be accessed by typing “help startF” on the MatLab command line. The next step in the preparation phase is to load the database into workspace. This may be accomplished in two ways. One way is load all of the database files from the original source. This is done by typing “startF” on the command line. The second option is an option which loads the database from previously saved and preprocessed binary files. This is accomplished by typing, “startC” on the command line in MatLab. The first option is used only when the user did not do any previous work in the Visualization Tool, or the work done was not saved. This option can be used if there is some suspicion that the database was in some way compromised. The first option loads the
database from the original files which takes much longer than using the startC option, thus making the startC option a more desirable option when possible. Both startC and startF are used only once in the beginning of the work. These functions can be used during the work as well, if there is a desire to restore the original database. All of the variables, which are loaded by these two functions, are used only for reading and Visualization Tool does not change the variables during the whole process. The next step involves using several other process and display functions. The functions which are available for use in this part of the research are searchUniv, triphones, words, and paramDisplay. These functions are used based on the users need for that particular function or action. The searchUniv function can be used when the user wants to display curves based on phones and additional prosodic features. The triphones function can be used when the user wants to display three consecutive phones in a curve. The words function can be used when the user wants to display the entire word in the curve. The paramDisplay function can be used when the user wants to display all nineteen parameters for a particular file. All of these functions are taking arguments and not returning anything. After the user states the request, the processing unit accesses the database binary or input. It then processes the data accordingly. The output has a display of the curves and preprocessed data in binary form. The system has the capability of storing viewed images. Images can be stored in different formats. The available formats are MatLab Figure, Adobe Illustrator file, Bitmap file, EPS file, Enhanced Metafile, JPEG Image, Paintbrush 24-bit file, Portable Bitmap file, Portable Document Format, Portable Graymap file, Portable Network Graphics file, Portable Pixmap file, Portable inKmap file, TIFF image, and TIFF no compression image. This is accomplished by pressing Control and S simultaneously while the figure window is active. After prompted, the user must choose the name of the figure as well as the desired format in which the file is to be saved. To finish
using the Visualization Tool, the user must type the command “finish” in the MatLab command window. This finish function will store all of the variables in the binary database and will clear the system memory.

2.1. **Prosodic Visualization**

Prosody has been visualized by displaying parameter curves. The visualization of parameter curves is imperative to the project. The curves have been analyzed in different ways in order to determine the prosodic impact on TalkingHead. The subheadings below map out the importance of specific curves and the way that they are displayed.

2.1.1. **Displaying of the Curve**

The project queried particular phonemes from the database and after acquiring corresponding parameters, it displays these parameters as a curve. This is accomplished by using the searchUniv function which takes the minimum of two parameters.

One example of a command that will achieve a search on ‘EH1’ which is looking for accented phone EH, in the word ‘marry’ with 100 percent curve extension, with open jaw parameter drawn in black, left centered, and with the possibility of adding extra curves in the next group will look like this: searchUniv(‘y’, ‘EH1’, ‘marry’, 100, 7, 0).

The first input parameter continues the work parameter. This parameter accepts character ‘n’ when plotted curves are the last curves to be displayed in a particular figure and ‘y’ if the user wants to add more curves to the current figure. Only the character ‘n’ clears the data and closes writing new curves to the particular figure. Character ‘y’ can be substituted with any other single character.

The second input parameter accepts a string value which must be written in single quotes. The string value is the name of the phone with eventual accent value as an added prosodic
feature. The rest of the values can be skipped and they are initialized to the default values. These values will be defined later in this chapter.

The third input parameter of the function is used to add extra prosodic feature to narrow the search. This parameter accepts string value in single quotes, which can be word, syllable, or parameter file name. If the user wishes to skip this parameter, then the user must enter "*" in order for this parameter to be bypassed.

The fourth input parameter of the function accepts numerical value, which is used for the extension of the curve. This value must be in a range between 0-100, where 0 is no extension and 100 is the full length of the phoneme extension. With the 100 range, the whole curve is three times bigger than the original.

The fifth input parameter accepts numerical value between 1-19 and the number corresponds with the parameter number from the parameter files. A list of corresponding parameters can be found by typing in the command window of MatLab, the command “help ParametersName”.

The sixth input parameter is the color parameter which accepts numeric value which ranges from 1-7 and these values are changing the color of the plotted curves.

The seventh input parameter accepts two numeric values either 0 or 1. This parameter is responsible for aligning the curves. The curves will be left aligned when 0 is used as the parameter and centered if 1 is used as the parameter.

The next three input parameters may be used to narrow the search with additional prosodic features and follow the same rules as the third input parameter. The searchUniv function will search the database for the inputted phone, with a narrowed search for extra prosodic features. It will retrieve the points from the parameter file, and the points will be
plotted on the plane and the curve will be fitted through the points and labeled accordingly. The curve will be distinctly labeled in order for it to be recognized. The curves are labeled using the file name that it came from, the word, and syllable. The labeling process becomes more important when dealing with multiple curves.

2.1.2. Displaying Many Curves

The outcome of the search decides how many curves will be displayed. If a narrowed search is conducted, then one curve will be displayed; however, if another, less narrowed, search function is conducted, many curves will automatically be displayed. In the same way that the single curves are displayed, the process is repeated for multiple curves as well. They too are labeled in order for easy user recognition. There is a special consideration for multiple curves which are used simultaneously. When many curves are displayed at the same time, a distinction is made between the curves in order to identify what prosodic features are connected to each curve.

![Figure 2.2 Multiple Curve Display Visualization.](image-url)
Figure 2.2 illustrates how the multiple curve distinction is achieved by numbering the curves. The legend shows how the number corresponds with each curve and what each curve contains.

2.1.3. Distinguishing Groups of Multiple Curves

As the curves are displayed and overlapped, the groups were distinguished by the color that was assigned to each group. This means that in addition to the processes discussed above, each group was assigned a color to distinguish it from the other groups.

Figure 2.3 Multiple Curve Groups Display Visualization.

Figure 2.3 illustrates how the groups are distinguished by color. One group of curves is displayed in green, while the other is displayed in red. The legend is also distinguished by color and is numbered to correspond with each curve and what each curve contains. This figure is designed only to show the ability of the system and is not meant to make a working case for comparison.
This made it easier for the user to differentiate the groups when the curves are used together. In addition, this made identifying patterns among groups easier and more efficient. The curves were assigned to a particular group based on their accent status such as no accent, primary accent, and secondary accent. Another grouping was based on prosodic features.

2.1.4. Curve Extension

Each curve has the possibility of expanding the starting and ending time and is user defined. Extending curves is important because the timing calculations could contain an error.

Figure 2.4 Curve Extensions.

Figure 2.4 illustrates a curve that has been extended to 100%. The ‘+’ indicates where the original, unexpended curve started and ended and is shown in the legend. In order to be more precise and make sure that the curve is correctly targeted, the timing was extended. This extension occurs on both sides, along the word curve. The user has control over
the extensions and the default was set to no extension. Another enhancing option is displaying triphones.

2.1.5. Triphones

The system has the possibility of displaying curves for three consecutive phonemes, or triphone, as one continuous curve.

Figure 2.5 TriPhone ‘M EH1 R’.

Figure 2.5 illustrates a triphone in which the triphone is labeled below the curve. The ‘+’ indicates where one phoneme ends and another begins.

The names of the phonemes used to draw the triphone curve, are in the label area underneath the curve portion for which they belong, for the functionality which was described earlier for regular curves. This option helps to see the possible changes in the phoneme curve caused by neighboring phonemes. The system also has the capability of comprehensively displaying parameters for individual phonemes.
2.1.6. Comprehensive Parameter Display

The system has the capability of displaying the whole set of parameters for a specific phoneme simultaneously.

Figure 2.6 All Parameters Display For A01.

Figure 2.6 illustrates all of the parameters for one particular file, in this case, A01. The file name is indicated in the title area located above the graph. This feature was used to compare each parameter to each other and extract any discrepancies or errors in parameter calculation. This feature of the project can also be very useful to help with optimization efforts. In addition to curve specifications and distinctions, there were particular requirements for the system design.

2.2. Requirements for System Design

This section focuses on the specific aspects of the system design. This points out what was done in order to make the system as user friendly and as efficient as possible. This helps to generate information faster and will contribute greatly to the outcome of this project as well as
future projects. The focus here is not on specific time requirements, but rather outlined what will need to be done. Future sections reveal how the elements of the system design was accomplished and how it can be verified.

2.2.1. User Friendly

The Visualization Tool is designed in three blocks for user friendly functionality which include initialization, work, and work completion. The initialization block allows the user to get into their work after completing a maximum of a two step initialization. This satisfies the proposal requirement that everything will be done with a maximum of five interactions from the beginning to the end. The work block allows the user to accomplish part of the work in a maximum of two steps which also satisfies the proposal requirement discussed above. The work completion block allows the user to complete the work in one step, which also fits within the proposal requirement guidelines. This allows for a more efficient process in realizing the endpoint. If the system was hard to use or slow, then the rest of the task might not come to fruition. The system also contains other possibilities which help the user to extract the desired information from the database.

2.2.2. Query Possibilities

The system contains query possibilities with the output of their addresses; however, they are concealed to the user. After the project was completed, there was no need to access any of the query information. If in the future a need will arise to access the queries, they can be accessed with one modification. The search is easy to use and helps make the task more efficient as well. It contains help that divulges what resources are available to search, such as tables, available phonemes, file names etc… Another important element of system design is the input formatting.
2.2.3. **Input Formatting**

The system is able to accept the database in Excel format and accepts the parameter files which were formatted in comma separated values. All of the input is automated, is done inside of the startF function, and is able to be accomplished in one step. Using the startF function, the user has the option of loading a new database and parameter files into the system. Using the startC function, the user also has the option of loading a preprocessed binary file which can be used for further processing. This will help keep all of the work current and up to date so that the work which has been completed will not have to be redone. The system also allows for display versatility.

2.2.4. **Display Versatility**

The system displays one or more curves which overlap in the same window and has the capability of displaying curves in different windows at the same time. If displaying in the same window, the curve requirements are used to achieve the requirements for this visualization. The system has the possibility for displaying the curves in different windows to be used for a lower number of curves in order to enhance the observation of differences. Another system feature is the straightforward user interface and convenient output.

2.2.5. **User Interface and Output**

The Visualization Tool has implemented a command line user interface which includes command line help files as well. The helpful user interface supports an equally helpful output possibility. All of the created figures are able to be saved in one of the following formats: MatLab Figure, Adobe Illustrator file, Bitmap file, EPS file, Enhanced Metafile, JPEG Image, Paintbrush 24-bit file, Portable Bitmap file, Portable Document Format, Portable Graymap file, Portable Network Graphics file, Portable Pixmap file, Portable inKmap file, TIFF image, or
TIFF no compression image. This allows for a portable and system independent output of any figures and the database.
3. VISUALIZATION TOOL DESIGN

This section provides a comprehensive description of how the project was brought to fulfillment. This section is also divided into two main sections: 1) Prosodic visualization and 2) System design; however, the focus is on the description of practical implementation of the requirements and the Visual Tool Design (see appendix A). The explanation of how the requirements were achieved, include the following: Displaying a single curve, displaying multiple curves, distinguishing groups of multiple curves, curve extensions, triphones, comprehensive parameter displays, system design, user friendly, query possibilities, input formatting, display versatility, and user interface and output. The following sections will give details about the system design. The explanation of the system design will be based on information that is the source code, which is available in Appendix D.

3.1. Implementation of Prosodic Visualization

The project had some challenges which needed to be overcome. The first challenge that the project encountered was MatLab limitations. The problem arose when switching the file name, which was provided in the database in a text format. This text should be used as a variable name; however, MatLab did not allow for this to occur. A special data structure had to be designed, to accommodate loading parameters into the variable using the text as parameter variable name. This was incorporated with a switch, which loaded parameters into paramvariable, based on the database text provided.

The second challenge arose during the testing phase, which showed that the system was too slow. After the initial learning curve for the software, the user became bored with answering the statements. The Visualization Tool had to be redesigned as a function that used input
parameters. This accommodated the speed requirement for the project; however, it posed another challenge.

MatLab uses separate workspaces for each function, which did not allow a function to load read only parameter variables in the switch. A special structure had to be designed in order to make these parameters available. Because these variables are accessed only for reading, they were available to functions as globals.

The remainder of this chapter explains the code design, as well as shows some use of the code. Contained within the chapter, are various instructions and examples.

3.1.1. Preparation for Displaying of the Curve

After the query (discussed below) of a particular phoneme is performed, the system grabs the particular phoneme pointer, which is located and accesses the parameters. These parameters are used to display a particular phoneme as a curve in the xyz-axis. The particular parameter is mapped into the xyz coordinates, using the parameter as the y value, and places index of the parameter as the x value. The z value is used to separate the curves in the 3D space. As indexes, the system uses integer from 1 to n where n is the last parameter which will be mapped. After mapping all of the points, the system function fits the function defined curve through the mapped points.

The particular curve has a display title and is labeled with information pertaining to what word, syllable, accent, and file, from which it originated. This is achieved inside of the searchUniv function, which uses input attributes to display desired curves. The input attributes of this function are explained in point 2.1.1 Displaying of the Curve.

The function searchUniv, contains two persistent variables. The first variable “leg” is used to construct the legend in the picture, and the second variable, “title1”, is used to construct
the title of the picture. Because MatLab creates variable workspace, which is separate for every function, access to the database files had to be linked from the main workspace to the function workspace. This was done by using the function available. The function uses ParametersName, to set up an array of the name of parameters. The names can be found in Appendix A.

The searchUniv function also initializes colors, as well as checks and initializes input arguments. The functions must have at least two input arguments. The rest of the input arguments can be initialized to default values if not specified on the command line. The default values are: for the extension of the curves is 0, for the parameter number is 1, for the color definition is 1, for the alignment argument is 0, and for prosody from 2-4 is ‘*’.

3.1.2. Implementation of Displaying Many Curves

The process for implementing the display of many curves was the same as the process for displaying a single curve. The difference is that when displaying many curves, each curve is mapped individually and displayed in xyz axis. In the case of multiple curves, a distinction is made using labeling. The title of the window states the name of the phoneme which is being mapped as multiple curves. The phone is queried by the query portion of the searchUniv function, which will be explained below in section 3.2.2. The outcome of the query portion of the function is stored as indexes of matching rows, from database in variable phoneSp. This variable is used to acquire the values from parameter files and build curves. First, the parameter variable is loaded into param variable, which is done by function paramSwitch. Next, the frame numbers are loaded. The starting frame of the curve is loaded into variable m and the end frame is loaded into variable n. This is achieved by getting the frame numbers from each indexed line in NUM database variable. After adding extensions to the curve, which is explained in section 3.1.4, the curve points are loaded into the curve variable. All of the curves are displayed using a
plot3 function. If the query brings back multiple curves then multiple curves are displayed as a group, and if the query brings back one curve, then only the single curve will be displayed.

3.1.3. Implementation of Distinguishing Groups of Multiple Curves

In the case of a multiple curves with different groups of phonemes, such as different accents, each accent group has a distinctive color assigned to it. The color differentiation is assigned only by group, not by the individual curve. This is accomplished by using parameter y as a first parameter in the searchUniv function. With this option, the new queried curves are added to the previous figure. To distinguish between groups, the parameter color in the searchUniv function should differ from those previously chosen. When using the y argument as the first argument of searchUniv, the user uses the advantage of consistent parameters as well as the hold on function. The hold on function is an internal function in MatLab.

3.1.4. Implementation of Curve Extension

Curve extensions were achieved by subtracting or adding an integer from the phoneme’s start time and end time respectively. The extension of the curve is user defined, by setting the extension variable to the desired value. This extension is calculated by taking the length of the phoneme and multiplying it by the extension variable and dividing it by 100. This value is added to the n and m values, which were discussed in the previously in section 3.1.2., and the new curve is calculated. The extension is marked by “+” the place for it is calculated before the curve is extended. The x coordinates for the “+” are held in the left_bar, right_bar variables. These markers are plotted using a plot3 function.

3.1.5. Implementation of Triphones

The implementation of triphones, uses the triphone function. If the user chooses to display the triphone rather than the phoneme by itself, two neighboring phonemes are added to
the curve by setting the start time to the start time of the proceeding phoneme in the database. To obtain this data, the starting index pointer is reduced by subtracting 1 and the finishing index pointer is increased by adding 1. Another possibility for this feature is getting values from the query which gives the user the direct starting and ending time.

3.1.6. Implementation of Comprehensive Parameter Display

The system has the capability of displaying the whole set of parameters for a specific parameter file simultaneously, by mapping all available parameters for a particular parameter file as ribbons. This implementation uses the ribbon function which is an internal function of MatLab. This feature extracts any discrepancies or errors in parameter calculation by visual examination.

3.2. Implementation of Requirements for System Design

This section focuses on the specific aspects of the system design. This segment discusses the various requirements being implemented in this part of the project. The focus here outlines how the elements of the Visualization Tool design were accomplished.

3.2.1. Implementation of a User Friendly System

The user friendly system was ensured by the incorporation of a help function. The other function that makes the system user friendly is the consolidation of commands that allows for fewer interactions. Every required action, which is specified in the requirements from the beginning to the end, including the query, was accomplished in no more than five steps. When accomplishing any of the requirements took more than five steps, then the system was modified by changing the design from prompt to command line functions.
3.2.2. **Implementation of Query Possibilities**

The system contains query possibilities with the output of their addresses by the query function which accepts text formatted parameters. These parameters determine what phoneme the user is searching for. One of the parameters provides a desired phoneme name. The names are based on the Carnegie Melon dictionary of phonemes which were derived from an earlier study [CMU 2007]. The second parameter defines the scope of the search and includes a file name. This file name is combined with restrictions for the desired output. This output has been piped into the file. The output has pointers to actual places where the phonemes are stored. The easy to use search contains a help function that divulges what resources are available from which to search, such as tables, available phonemes, file names etc... by calling a query help function which outputs all of this information. The next point discusses preparation of the input.

3.2.3. **Input Preparation**

The system is able to accept the database in Excel format and accepts the parameter files which were formatted in comma separated values. This was accomplished by a data preparation function which loads the database and parameters into the system as variables. The first function is startF which loads the database. The second function, startC handles loading the variables which were saved from the previous work in the prosody visualization system. The database in Excel had added preprocessed values for a simpler query handling. The calculation of frame numbers was moved from Visualization Tool to an Excel file, together with the calculation of the frequency of the phoneme.

3.2.4. **Implementation of Display Versatility**

The system displays one or more curves which are spaced out over the z axis, in the same window and have the capability of displaying curves in different windows at the same time.
This is accomplished in the function searchUniv, which allows a single window display. In the case of displaying a few curves, the system provides the functionality of displaying the curves in their own space.

3.2.5. Implementation of User Interface and Output

The system provides function help to implement an easy to use user interface. The output has zoom in and out, and rotation possibilities controlled by using the window menu and mouse. Saving the figures is provided as a pull down menu function and gives the user the possibility of saving the figure in the formats previously mentioned in Chapter 2.

3.3. Project Equipment Requirements

This project is able to run on any Von Newman architecture computer with either the Windows or Linux operating systems. For displaying curves, the computer must have a color monitor, and external storage capability. A color printer is desired, but optional.

3.4. Project Input Requirements

This project has two types of input files: database and set of parameters. The database file is in an Excel spreadsheet format. The data should be divided into columns as shown in Figure 3.1 (Phoneme, Phoneme Start Time, Phoneme Stop Time, Syllable, Syllable_Pitch_Accent, Word, Word Intonation Accent and File Name).
Figure 3.1 Sample Database File

All fields of the database have to be in text format, except for the starting and ending time fields of the phoneme, which are represented as single. Appendix E contains examples of database values and values of parameter files.

Figure 3.2 shows a sample parameter file. The file contains 20 columns with parameters.
The name of the parameter file must match the name of the sound file from which it was generated, and it has to have a “.param” extension. The first field must be an integer and state the frame number. The rest of the fields must be represented in double precision, in a range from zero to one, and match nineteen face muscle parameters in the given frame. Fields are comma separated and are described in Appendix C.
4. EVALUATION AND RESULTS

The testing phase of the project was conducted in two ways, by using both the black box, and glass box testing methods. In the black box testing phase, the tester was provided with pre-calculated data and performed tests based on the requirements for the project. This data came from the original files and used as an evaluation the first two sets of the data A and B which are supposed to be similar to one another. Set A and B, came from two experiments which were set up as control values. After the tester had completed the task, they compared the actual output with the provided desired output. This phase checked the correctness of the fundamental aspect of the system. The rest of the requirements were checked by trying to run the actual visualization project on the system. Visualization Tool was tested by 2 individuals and all of the problems which were identified were corrected. Based on the testing, the system went back to the design phase. The first version of the Visualization Tool used prompt type functions which were found to be inefficient in the research project because it drastically slowed down the users. The newly designed system changed to command line functions with input arguments in order to alleviate the problem. The first design was very sensitive to input errors. After the user made an input mistake, they had to rerun the search from the beginning. After the modification, the users made fewer mistakes because they used the capability of MatLab bringing up the last commands from the history. In the glass box testing phase, the tester examines the internal logical structure of the project. This is possible by using the up arrow key and modifying only the input values. Most of the input values stay the same for long periods of time; therefore the user only had to change two or three input values. This modification increased the user’s speed when conducting the research. The new system went back to the testing phase again and was tested by the same individuals. These individuals used both the Linux and Windows OS for testing. These tests
brought up another issue with the Linux system which did not want to accept the newer Excel files (Excel XP and above) without errors. This problem was corrected by loading in Linux the preprocessed binary file. The glass box testing was performed by visually examining the code and testing each particular function with predefined input. The expectation was that the tester will arrive at the predetermined result. After checking each function, the function interactions were checked for logical correctness. This part of the test used the bottom to the top approach.

![Figure 4.1 Curve Two Groups No Extension OW0 and OW1](image)

**Figure 4.1 Curve Two Groups No Extension OW0 and OW1**

Figure 4.1 illustrates the ability of the system to display two different groups of curves which are distinguished by color. In the title, different phone names for different groups can be seen. The first group is not accented and the second group contains an accent. Both parameters are parameters for Open Jaw. In the legend, the user can see the file which the curve came from.
as well as the syllable. The first group of the files came from po‘st and the second group came from the syllable po‘nt. The corresponding numbers and colors respond to the z axis, which is the curve number axis.

Figure 4.2 Curve Two Groups with Extension EH for A03 and B03

Figure 4.2 illustrates the ability of the Visualization Tool to display different curve groups which are distinguished by colors. It also shows the ability of the Tool to extend the curves to the amount specified by the user. The point of extension is marked by the “+” character on the curve. This ability can easily distinguish between the original curve and the added on portion. This example shows the ability of the Visualization Tool to narrow the search by the parameter file name.
Figure 4.3 illustrates two curve groups which came from a search of the word “combine” and “disobeyed”.

Figure 4.3 Curve Two Groups with Extension B
Figure 4.4 Curve Two Groups with Extension ER0, A04 and A07

Figure 4.5 illustrates the ability of the Visualization Tool to search and visualize different groups of curves which visualize the same phone from different parameter files.
Figure 4.5 Curve Two Groups with Extension EH1 word marry

Figure 4.6 illustrates the possibility of displaying different curves in one group. The curves are distinguished by the number along the z axis. Red curves are centered and green curves are not. This display shows the redundancy of two searches, which end up with the same shape of curves and shifted to one side.
Figure 4.6 shows the same phone and the same syllable from two different files. If extensions are not considered, the shape of the curve is the same, with the yellow curve value being larger than the red. This phone was extended 500% in order to compare it on a larger scale.
Figure 4.7 Curve Comparison from Different Files, Different Syllables and Aligned

Figure 4.7 shows a set of curves that are aligned with each other. Two different syllables are illustrated, which show similarities between the red and yellow curves and the green and blue curves respectively. This concludes that an increase in volume is an increase in parameters. This can be tested in all of the other syllables and if the same is true, then the conclusion can become a thesis for consideration.
Figure 4.8 Curve Comparison from Different Files, different syllable and not aligned

Figure 4.8 illustrates the exact search from figure 4.7 with curves that are not aligned. In this case, it is more difficult to make conclusions like the one made above.
Figure 4.9 All the length of the data display files 01, A, B, C and D.

Figure 4.9 shows four files in their entirety, rather than individual phonemes. This allows the files to be compared based on different prosodic features as a whole.
5. CONCLUSION AND FUTURE WORK

This project created a Visualization Tool, which allows a user to compare the linear and non-linear relationship between prosody and the change in shape by looking at the parameter values as a curve. The linear relationships are easy to find with stats; however, the non-linear relationships cannot be found in other ways, thus the need for the Visualization Tool. The Visualization Tool allows the user to explore data and query different information, such as, searching phones under different prosodic conditions. Exploring visual prosody allows for the potential advancement in TalkingHead research. This project was based on the proposal, which contained several requirements (Appendix A). These requirements were all successfully met and in some cases, surpassed. The Visualization Tool is user friendly and has been tested to ensure that any user encountered problems have been solved.

In the future, the Visualization Tool can be expanded in several ways: user interaction with curves, searches on word, GUI (graphical user interface), and displaying motion capture data together with the curves. The Visualization Tool can be expanded to allow more interactions with the graphic itself, based on the choices that the user makes. This could be done by attaching a menu to the mouse, which can be activated by a button, while hovering over an object. Another possibility is that a particular curve could be displayed based on mouse selection. An extra database could bring words themselves as well as the starting and ending times of the words. This would allow for the possibility of displaying entire words as curves. Another possibility for future expansion is to design GUI (graphical user interface), which could be a benefit for certain users. Lastly, while displaying the curve, animation can be incorporated to show the actual visual facial deformation.
The possibilities for future work are not limited to the ideas listed above; they are only a starting point for the Visualization Tool expansion. As expansion ideas are implemented, the possibilities for further expansions increase. As we continue to successfully eliminate the barriers that exist between humanity and technology, research results propel the need for more expansion and advancements in existing research as well.
ACKNOWLEDGEMENTS

I would like to acknowledge Dr. Scott King for allowing me to be involved in working on his TalkingHead. His participation in my academic pursuit has been unwavering and tireless. He has given me the utmost support in this research project and has helped with every phase, including testing.

I would also like to acknowledge Stan Leja, who provided additional academic support by providing his research for me to continue and who was also a tester for this project.

I would also like to extend my gratitude for receiving the following grants: Texas Research Development Fund Grant-Texas A&M CC, Texas Excellence Grant-Texas A&M CC, and NASA grant, for providing partial financial support for this project.

Finally, I would like to thank the Committee Chairperson: Dr. Scott King, and Committee Members: Dr. Hongyu Guo, and Dr. David Thomas, for their time and support in my academic endeavor.
BIBLIOGRAPHY AND REFERENCES


APPENDIX A

PROJECT REQUIREMENTS

1. Prosody was visualized by showing parameters as curves.

2. When many curves are displayed at the same time, a distinction was made between the curves in order to identify what prosodic features are connected to each curve.

3. As the curves are displayed and overlapped, the groups are distinguished by the color that was assigned to each group.

4. Each curve has the possibility of expanding the starting and ending time and will be possibly user defined.

5. The system has the possibility of searching for phonemes.

6. The system accepts the database in one of the known formats and accepts the parameter files which were formatted in comma separated values.

7. The system has the possibility of displaying curves for three consecutive phonemes or triphones as one continuous curve.

8. It displays one or more curves which overlap in the same window and has the capability of displaying curves in different windows at the same time.
9. The system has the capability of displaying the whole set of parameters for a specific phoneme simultaneously.

10. It implements an easy to use user interface and if it is a command line user interface then it includes command line help files as well.

11. All of the created figures are able to be saved in at least one known picture format.

12. The system has the capability for recording the history in order for the research to be easily replicated.

13. The system has the possibility of easy expansion (adding extra functionality does not require modifying any if the existing functions; however, it must have the capability of using them.)
APPENDIX B – PROSODY CORPUS

The current CMU phoneme set has 39 phonemes, not counting variant for lexical stress.

**Phoneme Example Translation**

<table>
<thead>
<tr>
<th>Phoneme</th>
<th>Example</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>odd</td>
<td>AA D</td>
</tr>
<tr>
<td>AE</td>
<td>at</td>
<td>AE T</td>
</tr>
<tr>
<td>AH</td>
<td>hut</td>
<td>HH AH T</td>
</tr>
<tr>
<td>AO</td>
<td>ought</td>
<td>AO T</td>
</tr>
<tr>
<td>AW</td>
<td>cow</td>
<td>K AW</td>
</tr>
<tr>
<td>AY</td>
<td>hide</td>
<td>HH AY D</td>
</tr>
<tr>
<td>B be</td>
<td>B IY</td>
<td></td>
</tr>
<tr>
<td>CH</td>
<td>cheese</td>
<td>CH IY Z</td>
</tr>
<tr>
<td>D dee</td>
<td>D IY</td>
<td></td>
</tr>
<tr>
<td>DH</td>
<td>thee</td>
<td>DH IY</td>
</tr>
<tr>
<td>EH</td>
<td>Ed</td>
<td>EH D</td>
</tr>
<tr>
<td>ER</td>
<td>hurt</td>
<td>HH ER T</td>
</tr>
<tr>
<td>EY</td>
<td>ate</td>
<td>EY T</td>
</tr>
<tr>
<td>F fee</td>
<td>F IY</td>
<td></td>
</tr>
<tr>
<td>G green</td>
<td>G R IY N</td>
<td></td>
</tr>
<tr>
<td>HH</td>
<td>he</td>
<td>HH IY</td>
</tr>
<tr>
<td>IH</td>
<td>it</td>
<td>IH T</td>
</tr>
<tr>
<td>IY</td>
<td>eat</td>
<td>IY T</td>
</tr>
<tr>
<td>JH</td>
<td>gee</td>
<td>JH IY</td>
</tr>
<tr>
<td>K key</td>
<td>K IY</td>
<td></td>
</tr>
<tr>
<td>L lee</td>
<td>L IY</td>
<td></td>
</tr>
<tr>
<td>M me</td>
<td>M IY</td>
<td></td>
</tr>
<tr>
<td>N knee</td>
<td>N IY</td>
<td></td>
</tr>
<tr>
<td>NG</td>
<td>ping</td>
<td>P IH NG</td>
</tr>
<tr>
<td>OW</td>
<td>oat</td>
<td>OW T</td>
</tr>
<tr>
<td>OY</td>
<td>toy</td>
<td>TOY</td>
</tr>
<tr>
<td>P pee</td>
<td>P IY</td>
<td></td>
</tr>
<tr>
<td>R read</td>
<td>R IY D</td>
<td></td>
</tr>
<tr>
<td>S sea</td>
<td>S IY</td>
<td></td>
</tr>
<tr>
<td>SH</td>
<td>she</td>
<td>SH IY</td>
</tr>
<tr>
<td>T tea</td>
<td>T IY</td>
<td></td>
</tr>
<tr>
<td>TH</td>
<td>theta</td>
<td>TH EY T AH</td>
</tr>
<tr>
<td>UH</td>
<td>hood</td>
<td>HH UH D</td>
</tr>
<tr>
<td>UW</td>
<td>two</td>
<td>T UW</td>
</tr>
<tr>
<td>V vee</td>
<td>V IY</td>
<td></td>
</tr>
<tr>
<td>W we</td>
<td>W IY</td>
<td></td>
</tr>
<tr>
<td>Y yield</td>
<td>Y IY L D</td>
<td></td>
</tr>
<tr>
<td>Z zee</td>
<td>Z IY</td>
<td></td>
</tr>
<tr>
<td>ZH</td>
<td>seizure</td>
<td>S IY ZH ER</td>
</tr>
</tbody>
</table>
APPENDIX C – Parameters Description

1. Frame Number
2. k0 = OPEN_JAW - opens jaw
3. k1 = JAW_IN - moves jaw in
4. k2 = Space for future use
5. k3 = ORB_ORIS - contracts lips, making mouth opening smaller
6. k4 = L_RIS, Left Risiorius - moves left corner towards ear
7. k5 = R_RIS, Right Risiorius - moves right corner towards ear
8. k6 = L_PLATYSMA, Left Platysma - moves left corner downward and lateral
9. k7 = R_PLATYSMA, Right Platysma - right corner downward and lateral
10. k8 = L_ZYG, Left Zygomaticus - raises corner up and lateral
11. k9 = R_ZYG, Right Zygomaticus - raises corner up and lateral
12. k10 = L_LEV_SUP - moves left top lip up
13. k11 = R_LEV_SUP - moves right top lip up
14. k12 = DEP_INF - opens both lips
15. k13 = Space for future use
16. k14 = MENTALIS - pulls lips together
17. k15 = L_BUCCINATOR - pulls back at left corner
18. k16 = R_BUCCINATOR - pulls back at right corner
19. k17 = INCISIVE_SUP - top lip rolls over bottom lip
20. k18 = INCISIVE_INF - bottom lip rolls over top lip
APPENDIX D – Source Code

Complete source code will be included on Media CD

Function searchUniv

function a=searchUniv(phone,prosody,extent,par_num,color_def,align_arg,prosody2,prosody3,prosody4)

% search Univ version 0.3
% i.e:
>>searchUniv2('EH1','marry',100,1,7,0)
% To use this function you have to type searchUniv2 followed by parameters:
% This function gets a maximum of 9 parameters
% A typed function should look like this:
% searchUniv2(phone,prosody,extension,parameter number,color,
% alignment,extra features X 3)
% i.e: >>searchUniv2('EH1','marry',100,2,1,0)
% to skip prosody put '*' as 2nd argument
% phone ex: 'EH1', prosody ex: 'marry',
% extension: number from 0-100,
% parameter number: 1-19, color number 1-7
% or length(COLOR_DEF),
% alignment: 0 for left and 1 for center
% alignment
% extra features: max of 3 extra features can
% be added, which will
% work in the function to narrow search.
% See also PARAMETERSNAME,
PHONEME, LOADFRESH
% i.e:
>>searchUniv2('EH1','marry',100,1,7,0)
% In this block I saved some parameters of
the function's workspace
% for future use by this function
persistent leg
persistent title1
available;
ParametersName; % set up name array with
names of parameters
% This block checks number of parameters
if nargin<1
    error('Not enough input arguments.')
else
    nargin ==1
        prosody="*";
        extent=0;
        par_num=1;
        color_def=1;
        align_arg = 0;
        prosody2="*";
        prosody3="*";
        prosody4="*";
    elseif nargin == 2
        extent=0;
        par_num=1;
        color_def=1;
        align_arg = 0;
        prosody2="*";
        prosody3="*";
        prosody4="*";
    elseif nargin == 3
        par_num=1;
        color_def=1;
        align_arg = 0;
        prosody2="*";

elseif nargin == 4
color_def = 1;
align_arg = 0;
prosody2 = '*';
prosody3 = '*';
prosody4 = '*';
elseif nargin == 5
align_arg = 0;
prosody2 = '*';
prosody3 = '*';
prosody4 = '*';
elseif nargin == 6
prosody2 = '*';
prosody3 = '*';
prosody4 = '*';
elseif nargin == 7
prosody3 = '*';
prosody4 = '*';
elseif nargin == 8
prosody4 = '*';
end

COLOR_DEF = [1 0 0; 0 1 0; 0 0 1; 1 0 1; 0 1 0; 0 0 0]; % definitions of colors
% To add more colors into COLOR_DEF, add RGB value after ';' inside square brackets

% This block checks range of the numeric values
if ~isnumeric(extent)
    error('extent should be a numeric value 0-100')
elseif extent < 0 & extent > 100
    error('extent should have a value 0-100')
end
if ~isnumeric(par_num)
    error('par_num should be a numeric value 1-19')
elseif par_num < 1 & par_num > 19
    error('par_num should have a value 1-19')
end
if ~isnumeric(color_def)
    error('color_def should be numeric value 1-7')
elseif color_def < 1 & color_def > length(COLOR_DEF)
    error('color_def should have a value 1-7')
end

if ~isnumeric(align_arg)
    error('align_arg should be numeric value 1-7')
elseif align_arg < 0 & align_arg > 1
    error('align_arg should have a value of 0 or 1')
end

temp_extent_mark = -1;

point = strmatch(phone, TXT(:, 1));
% search for phone: var point holds matched values of indexes from search
% above search is performed on first column of DB, search is looking for the phone
max = length(point);

if prosody == '*'
    % if there is no prosodic feature to search
    phoneSp = point;
else
    phoneSp = -1; % initializing
    % variable phoneSp holds indexes of search matching variables after
    % narrowing search with prosody
    for i = 1:max
        % This part searches for prosody in Phone query using a variable point as
        % the reference. Var point holds values of matching phone in rows from TXT
        % var 'i' makes search possible in query of phones. First value of
        % matching possibility is in TXT(first matching row for phone, from
% position 2 to the end of the row.
temp = strmatch(prosody, TXT(point(i,1),2:end));
if temp > 0
    % recording match row number
    temp_ph = point(i,1);

    % adding matching row number to the list as next row
    phoneSp = [phoneSp; temp_ph];
end
end

% discarding first row of -1 from initialization
phoneSp = phoneSp(2:end,:);

if prosody2 ~= '*'
    phoneSp1 = -1;
    for i = 1:length(phoneSp)
        % This part searches for prosody in Phone query using a variable phoneSp as the reference. Var 'i' makes search possible in query of phones. First value of % matching possibility is in TXT(first matching row for phone, from position 2 to the end of the row. temp = strmatch(prosody2,
        TXT(phoneSp(i,1),2:end)); if temp > 0
            % recording match row number
            temp_ph = phoneSp(i,1);

            % adding matching row number to the list as next row
            phoneSp1 = [phoneSp1; temp_ph];
        end
    end
end

% discarding first row of -1 from initialization
phoneSp = phoneSp1(2:end,:);

if prosody3 ~= '*'
    phonePh1 = -1;
    for i = 1:length(phoneSp)
        % This part searches for prosody in Phone query using a variable phoneSp as the reference. Var 'i' makes search possible in query of phones. First value of % matching possibility is in TXT(first matching row for phone, from position 2 to the end of the row. temp = strmatch(prosody3,
        TXT(phoneSp(i,1),2:end)); if temp > 0
            % recording match row number
            temp_ph = phoneSp(i,1);

            % adding matching row number to the list as next row
            phoneSp1 = [phoneSp1; temp_ph];
        end
    end
end

% discarding first row of -1 from initialization
phoneSp = phoneSp1(2:end,:);

if prosody4 ~= '*'
    phonePh1 = -1;
    for i = 1:length(phoneSp)
        % This part searches for prosody in Phone query using a variable phoneSp as the reference. Var 'i' makes search possible in query of phones. First value of % matching possibility is in TXT(first matching row for phone, from position 2 to the end of the row. temp = strmatch(prosody4,
        TXT(phoneSp(i,1),2:end)); if temp > 0
            % recording match row number
            temp_ph = phoneSp(i,1);

            % adding matching row number to the list as next row
            phoneSp1 = [phoneSp1; temp_ph];
        end
    end
end

end

end

end
phoneSp = phoneSp(2:end,:); end

max=length(phoneSp);
curve=zeros(2,1);

% block for displaying the curves for i=1:max

% getting parameter's name from matching row. Var phoneSp holds pointers to matching rows from the TXT, position 10 in the row holds parameter's name.
fname=char(TXT(phoneSp(i,1),10));

% paramSwitch gets fname and writes into workspace variable param which holds values of the parameters from fname
paramSwitch

% var m and n holds values for begining and ending frame of the phone respectively
m = NUM(phoneSp(i,1),1);
n = NUM(phoneSp(i,1),2);

% extension is used to extend phone curve extension=ceil((n-m)* extent/100);

% temp3 is used to calculate length of the phone for markings of the extension place
temp3=param(m:n,par_num+1);
phone_size=length(temp3);

% var m and n holds values for begining and ending frame of the phone respectively with the extension
m=m-extension;
n=n+extension;

temp2 is used for storing parameter values of current phone

x=length(temp2);

% left and right bar is used to store time value of the place of the extension
left_bar =((x-phone_size)/2);
right_bar = left_bar+phone_size;
curve = temp2;

% this block is used for left and center alignment of the curves
if align_arg == 0
    align_curves = 1:x; % left alignment
elseif align_arg == 1
    align_curves = ceil(-x/2):ceil(x/2)-1;
end

% center alignment

% plot3 is plotting curves one by one in 3D space
plot3(ones(x(1,1),1)*i,align_curves,curve,'Color',COLOR_DEF(color_def,:))

hold on % holding figure for future use

if length(leg)<1  % if string to be displayd for the legend is empty
    leg = [int2str(i) '. ''char(TXT(phoneSp(i,1),10)) ' Sylable: ' char(TXT(phoneSp(i,1),6))];
else
    leg1 = [int2str(i) '. ''char(TXT(phoneSp(i,1),10)) ' Sylable: ' char(TXT(phoneSp(i,1),6))];

    leg=strvcat(leg,leg1); % adding aditional row of text to be displayed

end
if extent > 0

plot3(i,align_curves(left_bar),curve(left_bar),'k+')

plot3(i,align_curves(right_bar),curve(right_bar),'k+')
    leg=strvcat(leg,'extension','extension');
    if length(temp_extent_mark) == 1

    temp_extent_mark=[i,align_curves(left_bar),curve(left_bar);i,align_curves(right_bar),curve(right_bar)];
    else

    temp_extent_mark=[temp_extent_mark;i,align_curves(left_bar),curve(left_bar);i,align_curves(right_bar),curve(right_bar)];
    end

    end

ylabel('TIME/index')
zlabel('PARAMETER VALUE')
xlabel('CURVE NUMBER')

%displaying legend
legend(leg)

end

grid on the displayed figure
grid on

if length(title1)<1 %if string to be displayd for the title is empty

    title1 = ['phone: '  char(phone) '; Extra search: ' char(prosody) ' - parameter: ' paramName(par_num,:)];
else
    title2 = ['phone: '  char(phone) '; Extra search: ' char(prosody) ' - parameter: ' paramName(par_num,:)];

    title1=strvcat(title1,title2); %adding aditional row of text to be displayed
end

title(title1) %displaying figure's title

contin = input('Would you like to add additional curves?  y or n > ','s');

if contin == 'n' %if string to be displayd for the title is empty
    hold off
    clear leg
end
Function loadFresh

%loadFresh.m version 1.0 (6-7-07)
%function loads fresh data into system
%after loadFresh, verification can be done
%by typing on command line >> whos
%clear all
%create access to the variables
n=300; %control value for rows to extend matrix
m=20; %control value for columns to extend matrix
[NUM,TXT,RAW]=xlsread('DBPhoneme.xls'); %load excel DB into: NUM - Numeric array, TXT - text array, RAW - cell array
A01=csvread('styleA01.parm'); %load parameters into corresponding array (numeric); name convention: stale*.parm -> *
A01=[A01; zeros(n, m)]; %extending array with zeros
A02=csvread('styleA02.parm');
A02=[A02; zeros(n, m)];
A03=csvread('styleA03.parm');
A03=[A03; zeros(n, m)];
A04=csvread('styleA04.parm');
A04=[A04; zeros(n, m)];
A05=csvread('styleA05.parm');
A05=[A05; zeros(n, m)];
A06=csvread('styleA06.parm');
A06=[A06; zeros(n, m)];
A07=csvread('styleA07.parm');
A07=[A07; zeros(n, m)];
A08=csvread('styleA08.parm');
A08=[A08; zeros(n, m)];
A09=csvread('styleA09.parm');
A09=[A09; zeros(n, m)];
A10=csvread('styleA10.parm');
A10=[A10; zeros(n, m)];
A11=csvread('styleA11.parm');
A11=[A11; zeros(n, m)];
A12=csvread('styleA12.parm');
A12=[A12; zeros(n, m)];
A13=csvread('styleA13.parm');
A13=[A13; zeros(n, m)];
A14=csvread('styleA14.parm');
A14=[A14; zeros(n, m)];
A15=csvread('styleA15.parm');
A15=[A15; zeros(n, m)];
A16=csvread('styleA16.parm');
A16=[A16; zeros(n, m)];
B01=csvread('styleB01.parm');
B01=[B01; zeros(n, m)];
B02=csvread('styleB02.parm');
B02=[B02; zeros(n, m)];
B03=csvread('styleB03.parm');
B03=[B03; zeros(n, m)];
B04=csvread('styleB04.parm');
B04=[B04; zeros(n, m)];
B05=csvread('styleB05.parm');
B05=[B05; zeros(n, m)];
B06=csvread('styleB06.parm');
B06=[B06; zeros(n, m)];
B07=csvread('styleB07.parm');
B07=[B07; zeros(n, m)];
B08=csvread('styleB08.parm');
B08=[B08; zeros(n, m)];
B09=csvread('styleB09.parm');
B09=[B09; zeros(n, m)];
B10=csvread('styleB10.parm');
B10=[B10; zeros(n, m)];
B11=csvread('styleB11.parm');
B11=[B11; zeros(n, m)];
B12=csvread('styleB12.parm');
B12=[B12; zeros(n, m)];
B13=csvread('styleB13.parm');
B13=[B13; zeros(n, m)];
B14=csvread('styleB14.parm');
B14=[B14; zeros(n, m)];
B15=csvread('styleB15.parm');
B15=[B15; zeros(n, m)];
B16=csvread('styleB16.parm');
B16=[B16; zeros(n, m)];
C01=csvread('styleC01.parm');
C01=[C01; zeros(n, m)];
C02=csvread('styleC02.parm');
C02=[C02; zeros(n, m)];
C03=csvread('styleC03.parm');
C03=[C03; zeros(n, m)];
C04=csvread('styleC04.parm');
C04=[C04; zeros(n, m)];
C05=csvread('styleC05.parm');
C05=[C05; zeros(n, m)];
C06=csvread('styleC06.parm');
C06=[C06; zeros(n, m)];
C07=csvread('styleC07.parm');
C07=[C07; zeros(n, m)];
C08=csvread('styleC08.parm');
C08=[C08; zeros(n, m)];
C09=csvread('styleC09.parm');
C09=[C09; zeros(n, m)];
C10=csvread('styleC10.parm');
C10=[C10; zeros(n, m)];
C11=csvread('styleC11.parm');
C11=[C11; zeros(n, m)];
C12=csvread('styleC12.parm');
C12=[C12; zeros(n, m)];
C13=csvread('styleC13.parm');
C13=[C13; zeros(n, m)];
C14=csvread('styleC14.parm');
C14=[C14; zeros(n, m)];
C15=csvread('styleC15.parm');
C15=[C15; zeros(n, m)];
C16=csvread('styleC16.parm');
C16=[C16; zeros(n, m)];
D01=csvread('styleD01.parm');
D01=[D01; zeros(n, m)];
D02=csvread('styleD02.parm');
D02=[D02; zeros(n, m)];
D03=csvread('styleD03.parm');
D03=[D03; zeros(n, m)];
D04=csvread('styleD04.parm');
D04=[D04; zeros(n, m)];
D05=csvread('styleD05.parm');
D05=[D05; zeros(n, m)];
D06=csvread('styleD06.parm');
D06=[D06; zeros(n, m)];
D07=csvread('styleD07.parm');
D07=[D07; zeros(n, m)];
D08=csvread('styleD08.parm');
D08=[D08; zeros(n, m)];
D09=csvread('styleD09.parm');
D09=[D09; zeros(n, m)];
D10=csvread('styleD10.parm');
D10=[D10; zeros(n, m)];
D11=csvread('styleD11.parm');
D11=[D11; zeros(n, m)];
D12=csvread('styleD12.parm');
D12=[D12; zeros(n, m)];
D13=csvread('styleD13.parm');
D13=[D13; zeros(n, m)];
D14=csvread('styleD14.parm');
D14=[D14; zeros(n, m)];
D15=csvread('styleD15.parm');
D15=[D15; zeros(n, m)];
D16=csvread('styleD16.parm');
D16=[D16; zeros(n, m)];
E01=csvread('styleE01.parm');
E01=[E01; zeros(n, m)];
E02=csvread('styleE02.parm');
E02=[E02; zeros(n, m)];
E03=csvread('styleE03.parm');
E03=[E03; zeros(n, m)];
E04=csvread('styleE04.parm');
E04=[E04; zeros(n, m)];
E05=csvread('styleE05.parm');
E05=[E05; zeros(n, m)];
E06=csvread('styleE06.parm');
E06=[E06; zeros(n, m)];
E07=csvread('styleE07.parm');
E07=[E07; zeros(n, m)];
E08=csvread('styleE08.parm');
E08=[E08; zeros(n, m)];
E09=csvread('styleE09.parm');
E09=[E09; zeros(n, m)];
E10=csvread('styleE10.parm');
E10=[E10; zeros(n, m)];
E11=csvread('styleE11.parm');
E11=[E11; zeros(n, m)];
E12=csvread('styleE12.parm');
E12=[E12; zeros(n, m)];
E13=csvread('styleE13.parm');
E13=[E13; zeros(n, m)];
E14=csvread('styleE14.parm');
E14=[E14; zeros(n, m)];
E15=csvread('styleE15.parm');
E15=[E15; zeros(n, m)];
E16=csvread('styleE16.parm');
E16 = [E16; zeros(n, m)];
F01 = csvread('styleF01.parm');
F01 = [F01; zeros(n, m)];
F02 = csvread('styleF02.parm');
F02 = [F02; zeros(n, m)];
F03 = csvread('styleF03.parm');
F03 = [F03; zeros(n, m)];
F04 = csvread('styleF04.parm');
F04 = [F04; zeros(n, m)];
F05 = csvread('styleF05.parm');
F05 = [F05; zeros(n, m)];
F06 = csvread('styleF06.parm');
F06 = [F06; zeros(n, m)];
F07 = csvread('styleF07.parm');
F07 = [F07; zeros(n, m)];
F08 = csvread('styleF08.parm');
F08 = [F08; zeros(n, m)];
F09 = csvread('styleF09.parm');
F09 = [F09; zeros(n, m)];
F10 = csvread('styleF10.parm');
F10 = [F10; zeros(n, m)];
F11 = csvread('styleF11.parm');
F11 = [F11; zeros(n, m)];
F12 = csvread('styleF12.parm');
F12 = [F12; zeros(n, m)];
F13 = csvread('styleF13.parm');
F13 = [F13; zeros(n, m)];
F14 = csvread('styleF14.parm');
F14 = [F14; zeros(n, m)];
F15 = csvread('styleF15.parm');
F15 = [F15; zeros(n, m)];
F16 = csvread('styleF16.parm');
F16 = [F16; zeros(n, m)];
NUM = [0 0; NUM(:,3:4)];  % adding first row of zeros to match TXT or RAW indexing
clear m  % clear temporary m
clear n  % clear temporary n
% who % summary of variables
Function paramSwitch

%paramSwitch.m version 1.0 (6/09/07)
%This is a script file
%This file is using a switch to load the parameter file specified in variable
%'fname' into variable param. This is a script file which is using caller
%workspace to get 'fname' and pass 'param'.

switch fname
    case 'A01'
        param = A01;
    case 'A02'
        param = A02;
    case 'A03'
        param = A03;
    case 'A04'
        param = A04;
    case 'A05'
        param = A05;
    case 'A06'
        param = A06;
    case 'A07'
        param = A07;
    case 'A08'
        param = A08;
    case 'A09'
        param = A09;
    case 'A10'
        param = A10;
    case 'A11'
        param = A11;
    case 'A12'
        param = A12;
    case 'A13'
        param = A13;
    case 'A14'
        param = A14;
    case 'A15'
        param = A15;
    case 'A16'
        param = A16;
    case 'B01'
        param = B01;
    case 'B02'
        param = B02;
    case 'B03'
        param = B03;
    case 'B04'
        param = B04;
    case 'B05'
        param = B05;
    case 'B06'
        param = B06;
    case 'B07'
        param = B07;
    case 'B08'
        param = B08;
    case 'B09'
        param = B09;
    case 'B10'
        param = B10;
    case 'B11'
        param = B11;
    case 'B12'
        param = B12;
    case 'B13'
        param = B13;
    case 'B14'
        param = B14;
    case 'B15'
        param = B15;
    case 'B16'
        param = B16;
    case 'C01'
        param = C01;
    case 'C02'
        param = C02;
    case 'C03'
        param = C03;
    case 'C04'
        param = C04;
    case 'C05'
        param = C05;
    case 'C06'
        param = C06;
    case 'C07'
        param = C07;
case 'C08'    
    param = C08;

case 'C09'    
    param = C09;

case 'C10'    
    param = C10;

case 'C11'    
    param = C11;

case 'C12'    
    param = C12;

case 'C13'    
    param = C13;

case 'C14'    
    param = C14;

case 'C15'    
    param = C15;

case 'C16'    
    param = C16;

case 'D01'    
    param = D01;

case 'D02'    
    param = D02;

case 'D03'    
    param = D03;

case 'D04'    
    param = D04;

case 'D05'    
    param = D05;

case 'D06'    
    param = D06;

case 'D07'    
    param = D07;

case 'D08'    
    param = D08;

case 'D09'    
    param = D09;

case 'D10'    
    param = D10;

case 'D11'    
    param = D11;

case 'D12'    
    param = D12;

case 'D13'    
    param = D13;

case 'D14'    
    param = D14;

case 'D15'    
    param = D15;

case 'D16'    
    param = D16;

case 'E01'    
    param = E01;

case 'E02'    
    param = E02;

case 'E03'    
    param = E03;

case 'E04'    
    param = E04;

case 'E05'    
    param = E05;

case 'E06'    
    param = E06;

case 'E07'    
    param = E07;

case 'E08'    
    param = E08;

case 'E09'    
    param = E09;

case 'E10'    
    param = E10;

case 'E11'    
    param = E11;

case 'E12'    
    param = E12;

case 'E13'    
    param = E13;

case 'E14'    
    param = E14;

case 'E15'    
    param = E15;

case 'E16'    
    param = E16;

case 'F01'    
    param = F01;

case 'F02'    
    param = F02;

case 'F03'    
    param = F03;

case 'F04'    
    param = F04;

case 'F05'    
    param = F05;
case 'F06'
    param = F06;
case 'F07'
    param = F07;
case 'F08'
    param = F08;
case 'F09'
    param = F09;
case 'F10'
    param = F10;
case 'F11'
    param = F11;
case 'F12'
    param = F12;
otherwise
    fprintf(2,'Error: parameters for %s not found', fname);
    param = nan;
end

    case 'F13'
        param = F13;
    case 'F14'
        param = F14;
    case 'F15'
        param = F15;
    case 'F16'
        param = F16;
otherwise
    fprintf(2,'Error: parameters for %s not found', fname);
    param = nan;
end
Function PHONEME

% The current CMU phoneme set has 39 phonemes, not counting varia for lexical stress.
% Phoneme Example Translation
% --------- ---------  ---------
% AA  odd  AA  D
% AE  at  AE  T
% AH  hut  HH  AH  T
% AO  ought  AO  AH  T
% AW  cow  K  AW
% AY  hide  HH  AH  D
% B  be  B  IY
% CH  cheese  CH  IY  Z
% D  dee  D  IY
% DH  thee  DH  IY
% EH  Ed  EH  D
% ER  hurt  HH  ER  T
% EY  ate  HH  ER  T
% F  fee  F  IY
% G  green  G  R  IY  N
% HH  he  HH  IY
% IH  it  IHT
% IY  eat  IY  T
% JH  gee  JH  IY
% K  key  K  IY
% L  lee  L  IY
% M  me  M  IY
% N  knee  N  IY
% NG  ping  P  IH  NG
% OW  oat  OW  T
% OY  toy  T  OY
% P  pee  P  IY
% R  read  R  IY  D
% S  sea  S  IY
% SH  she  SH  IY
% T  tea  T  IY
% TH  theta  TH  EY  T  AH
% UH  hood  HH  UH  D
% UW  two  T  UW
% V  vee  V  IY
% W  we  W  IY
% Y  yield  Y  IY  L  D
% Z  zee  Z  IY
% ZH  seizure  S  IY  ZH  ER
%  
%}
Function Parameters

% 1. k0 = OPEN_JAW - opens jaw
% 2. k1 = JAW_IN - moves jaw in
% 3. k2 = Space for future use
% 4. k3 = ORB_ORIS - contracts lips, making mouth opening smaller
% 5. k4 = L_RIS, Left Risorius - moves left corner towards ear
% 6. k5 = R_RIS, Right Risorius - moves right corner towards ear
% 7. k6 = L_PLATYSMA, Left Platysma - moves left corner downward and lateral
% 8. k7 = R_PLATYSMA, Right Platysma - right corner downward and lateral
% 9. k8 = L_ZYG, Left Zygomaticus - raises corner up and lateral
% 10. k9 = R_ZYG, Right Zygomaticus - raises corner up and lateral
% 11. k10 = L_LEV_SUP - moves left top lip up
% 12. k11 = R_LEV_SUP - moves right top lip up
% 13. k12 = DEP_INF - opens both lips
% 14. k13 = Space for future use
% 15. k14 = MENTALIS - pulls lips together
% 16. k15 = L_BUCCINATOR - pulls back at left corner
% 17. k16 = R_BUCCINATOR - pulls back at right corner
% 18. k17 = INCISIVE_SUP - top lip rolls over bottom lip
% 19. k18 = INCISIVE_INF - bottom lip rolls over top lip

paramName=strvcat('OPEN_JAW', 'JAW_IN', 'Space for future use', 'ORB_ORIS', 'Left Risorius', 'Right Risorius', 'L_PLATYSMA', 'R_PLATYSMA', 'L_ZYG', 'R_ZYG', 'L_LEV_SUP', 'R_LEV_SUP', 'DEP_INF', 'Space for future use', 'MENTALIS', 'L_BUCCINATOR', 'R_BUCCINATOR', 'INCISIVE_SUP', 'INCISIVE_INF');
%This script is designed to make variables available in the workspace of
%the functions, and these files are accessed only for reading
%
global A01 A07 A13 B03 B09 B15 C05 C11 D01 D07 D13 E03 E09 E15 F05 F11 NUM
global A02 A08 A14 B04 B10 B16 C06 C12 D02 D08 D14 E04 E10 E16 F06 F12 RAW
global A03 A09 A15 B05 B11 C01 C07 C13 D03 D09 D15 E05 E11 F01 F07 F13 TXT
global A04 A10 A16 B06 B12 C02 C08 C14 D04 D10 D16 E06 E12 F02 F08 F14
global A05 A11 B01 B07 B13 C03 C09 C15 D05 D11 E01 E07 E13 F03 F09 F15
global A06 A12 B02 B08 B14 C04 C10 C16 D06 D12 E02 E08 E14 F04 F10 F16