A SIMPLE PROGRAM FOR THE VISUALISATION OF F0

Cioni Lorenzo

Laboratorio di Linguistica, Scuola Normale Superiore

ABSTRACT - The main aim of the present paper is to present a simple program (1) that has been developed at our Laboratory and the name of which is Pit. The scope of such a program ranges from visualisation to editing of F0. Pit, indeed, allows a user to visualise up to six F0 graphs within a single window and to perform over each of them both editing operations (such as clear or cut), measurements, justification and splitting.

INTRODUCTION

The main aim of this paper is to present a simple program that has been developed at our Laboratory and the name of which is Pit (Cioni, 1993). The scope of such a program ranges from visualisation to editing of F0. Pit, indeed, allows a user to visualise up to six F0 graphs within a single window and to perform over each of them both editing operations (such as *clear* or *cut*), measurements, splitting and justification.

Visualisation can be performed on a time vs. frequency diagram or on a percentage duration vs. frequency diagram: in the former case we use an absolute scale so to visualise pitches with their effective length (we use a logarithmic scale with the frequency in hertz and time in milliseconds) whereas in the latter case we allow a comparison of pitches of different lengths by displaying them as if they had the same length. In this case the longer pitches are compressed so to be displayed together with the shorter ones (see figure 1).

As to measurements Pit allows the user to perform simple measures over each pitch and to save the measured values within text files so to allow subsequent statistical computations.

Editing operations are very simple mainly because they have been planned to allow a user to remove unwanted F0 values (*clear*) or to cut the length of a pitch by eliminating a part of the graph. Among the editing operations we have the possibility to perform splitting operations, i. e. to fix one or more markers on a pitch graph so to split it in two or more parts.

When up to six pitches are displayed within a single window, they can be qualitatively compared among themselves but can be even justified by fixing a marker on each F0 so to allow a comparison of their behaviours starting from speech dependent positions.

The present application is to be considered either as a stand-alone module or as a module to be inserted within either a graph or a pipeline of co-operating applications of which it represents a terminal node. As to the pipeline we have an acquisition module (producing speech files), a module that performs analysis and extract F0 according to some algorithm (producing pitch files) and Pit that displays the result of such analysis: its nature of terminal module prevents Pit from producing new pitch files so to maintain a full consistency between a speech file and the corresponding pitch files.

A BRIEF OVERVIEW OF THE PROGRAM

Pit is a very simple and special purpose application with a clean and direct interface that is characterised by windows and buttons. Its being a special purpose application follows from the presence of a well defined and bounded set of operations that defines a precise domain: visualisation and editing.

The aforesaid characteristic derives from the need of implementing an application that is easy to use, is characterised by a well defined application area but can co-operate with other applications as well.

On the other hand, the cleanness and directness of the interface follow from the fact that each operation is associated in some way with a button.

In this way the set of buttons represents, at any moment, the set of available operations though some of them may need some precondition be satisfied.

Windows are used both for visualisation purposes and to interact with the user so to acquire values that are required by the application itself (dialog windows). So to avoid an excessive crowding of the screen, windows are either temporary or are shrunk to icons whenever they are unnecessary.

The application, besides a clean and direct interface, is characterised by a simple inner structure so that, even on an old style hardware such as a Dec Microvax, its performances are quite good. Such inner structure is composed by a set of nested loops and a set of procedures: during each loop the program waits for the user to select an object or press a button whereas each procedure is executed so to perform the task that corresponds to the selected button. Each task can concern either the application as a whole (buttons *start* and *quit*, with an obvious meaning) or one or more objects. With the term object we denote any passive component of the application, such as a window, one or more graphs or even a set of measured values.

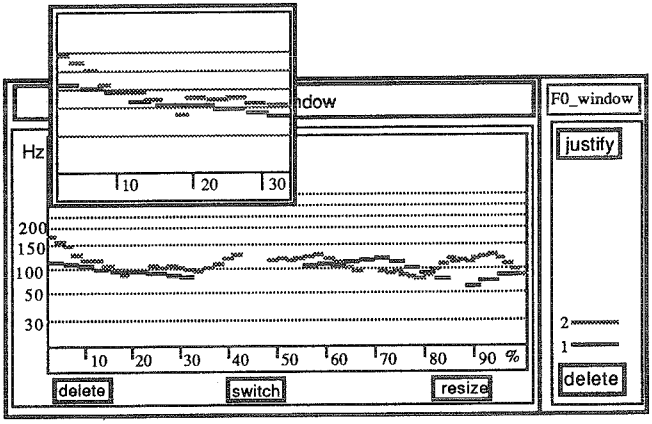

Figure 1.Visualisation.

VISUALISATION

The starting point is the visualisation of a bounded set of pitches. Therefore we have a set of pitch files that represent the outcome of some other application (for instance Audio, see figure 5) and among which the user can choose those to be displayed. Visualisation comes after a loading cycle that is characterised by the interaction with dialog windows for the selection of non interactive editing operations and of the type of the files to be loaded. The former allow the discarding of zero values at the beginning and/or the end of each file so that the comparison can be performed over meaningful values only whereas the latter allows the selection of pitch files of the same/different types so to allow even a comparison of the result of distinct algorithms over the same speech file (see figure 5).

After the loading phase, the visualisation of the selected pitches (2) is performed on a dedicated window (visualisation window, see figure 1) on which they are represented by using up to six distinct colours. This window is characterised by three buttons: *delete*, *switch* and *resize* (3): the selection of *delete* allows, at this stage, the deletion of the current windows and the selection of a new set of pitches, the selection of *switch* allows the switching between the two distinct styles of representation (from time vs. frequency to percentage duration vs. frequency and vice versa, see the introduction) whereas the selection of *resize* allows the changing of the size of the visualisation window (4).

Besides such a visualisation window we have two auxiliary windows (see figure 1). The former can be created/deleted by simply clicking the mouse middle button over the graphs and acts as a magnifier. It can be even moved towards either left or right so to allow a closer examination and a more exact qualitative comparison of small portions of the graphs. The latter (named selection window) contains two buttons and a list of names of the displayed pitches (5). The buttons are labelled *delete* and *justify* whereas the list lets the user select one of the F0 for either measurements (left mouse button) or editing (right mouse button). The *delete* button allows the deletion of the selection window (such window can be restored through a *restore* button that is created for this purpose only) so to allow a

greater resizing of the visualisation window whereas the *justify* button allows the justification of the F0s. Each of these operations (editing, measurements and justify) requires one or more dedicated windows and defines a new set of operations/buttons.

With the term justify (see figure 2) we mean the possibility for the user to fix on each F0 a marker so to align each graph on this marker. The markers are fixed one independently from the others. This operation is very useful since it allows the investigation of the behaviour of the displayed pitches starting from speech dependent positions. By justifying the pitches it is possible to examine the behaviour of the F0 of a key word within distinct contexts.

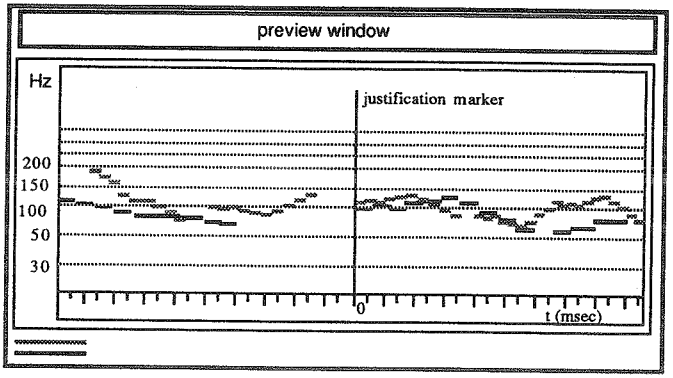

Figure 2.Justification.

The execution of the justification operation requires a set of buttons and a set of windows. The latter is composed by one window to visualise each F0, another to visualise the justified F0s and the last one that acts as a magnifier whenever the pitches have been justified. The set of buttons contains the following items: *back <*, *forw >*, *fix ><*, *preview*, *skip ->*, *done*, *abort* and *switch*. The meaning of each button should be clear enough so here we restrict ourselves to few remarks. The buttons *back <* and *forw >* let a user look through, backward and forward, the pitches so to be able to fix a marker on a F0 (*fix ><*) or to skip it (*skip ->*). The skipping of an F0 means that such a graph is not visualised at the end of the justification phase. The other buttons let the user interrupt the operation (*abort*) or examine what he/she has done so to continue (*preview*) or to stop (*done*).

As to measurements (see figure 3), Pit defines a set of correlated windows each one dedicated to a well defined task. The starting point is again the selection of one of the F0s and its visualisation on a window. We have, therefore, one window for the visualisation of a single pitch (measure window), one for the visualisation of the early twenty frequency values (data window), one that contains a set of function-labelled buttons (tools window) and one (history window) that contains the history of the outcomes of one of these functions (slope, see figure 3). As to the measure window, it displays a single F0 and allows the user to execute measures of pairs (time - frequency) with or without fixing a labelled cursor on the graph. Anyway, the frequency values appear even on the data window. We note that the measure window and the data window are always present simultaneously: the latter contains both the values of frequency and a button labelled *<<tools>>* whose selection causes the creation/deletion of the tools window.

This windows is characterised by a set of functional buttons such as *average* (it evaluates the mean frequency value), *max* (it highlights the maximum frequency value), *min* (it highlights the minimum frequency value), *slope* (it evaluates the ratio between an absolute difference of frequency and the corresponding of time and put the outcome on the history window), *history* (it creates/destroys the history window) and *list* (it allows the listing of frequency-time pairs between two markers and either their printing or their storing in text files).

EDITING

Editing assumes the selection of an F0 and gives rise to the creation of a new dedicated window and the shrinking of the existing ones. Pit is characterised by a limited set of classical editing operations

(such as *clear*, *cut*, *undo*, *close* and *save*) in addition to an operation that allows the switching between the aforesaid visualisation modes (*switch*) and one that is associated to a button labelled *split*.

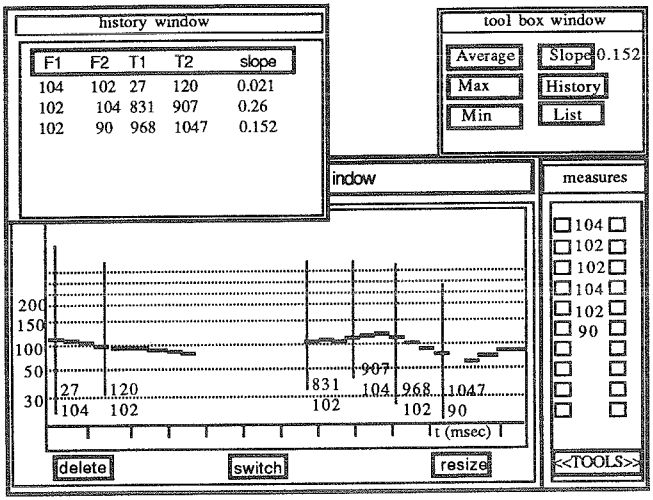

Figure 3. Measurements. Slope= | F2 - F1 | / | T2 - T1 |, sign from the graph.

This fact corresponds to a precise choice since Pit is mainly a visualisation program and therefore it accepts, as input, files that are the outcomes of other applications (the producers, for instance Audio, see figure 5) and so behaves as a consumer (see below and (Cioni, 1994)).

Editing operations are, therefore, very simple because they have been planned to allow a user to remove unwanted F0 values (*clear*) or to *cut* the length of a pitch by eliminating a part of the graph. The presence of the *undo* button allows the user to recover from any unwanted editing operation. The presence of the *save* button allows him/her to modify the values stored in main memory so that the modified graph can be more easily compared with the others.

There is no way, indeed, for a user to save the modified graph within a new or an existing file. The main reason for this fact lies in the aforesaid nature of Pit and on the requirement of maintaining a full consistency between a speech file and the corresponding pitch file.

The available classical editing operations (*clear* and *cut*) require the explicit selection of a part of a graph and then the pressing of the proper button. The main difference between *clear* and *cut* is that with a clear operation the length of the graph is unchanged though the selected part is cleared whereas a cut operation turns into the shortening of the corresponding graph. A positive side effect of a *cut* operation is that, by shortening a graph, it obtains a magnification of the graph itself, since the shortened graph is displayed on a window whose width is unchanged.

Splitting assumes the fixing of a marker in the selected position on the graph. After this fixing, the selection of the button labelled *split* turns into the breaking of the graph in two parts and their justification on the origin (see figure 4). Such operation can be performed recursively so to justify the words within a phrase (6) at the origin and compare the behaviour of the pitch over each of them.

GENERAL REMARKS ABOUT THE PROGRAM

The present application has been developed in order both to meet the requirements of the researchers that frequent the Laboratory and to co-operate with other applications that we already had.

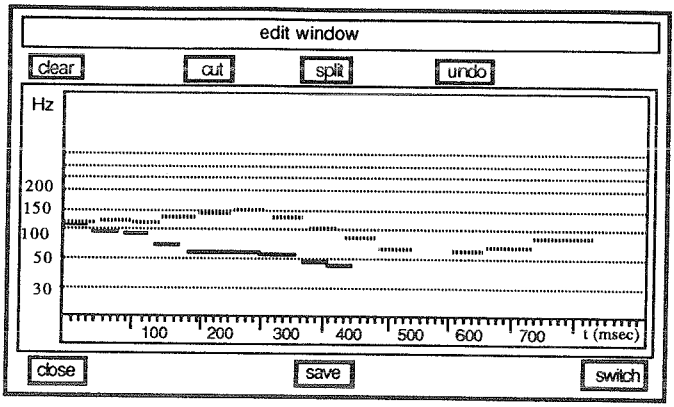

Figure4. Splitting.

From this point of view, the present program is to be considered both a stand alone program and, preferably, a module to be inserted either within a graph or within a pipeline of applications of which it represents a terminal node (see below).

As to the pipeline (see figure 5), for the time being, we have an application (Audio) that contains an acquisition module (that produces speech files) and two modules that perform analysis and extract F0 according to two distinct algorithms, so producing pitch files. Such algorithms are called Sift and Reetz.

The former is an LPC based algorithm that is more suited for male voices whereas the latter is well suited also for female voices.

As to Sift, it performs speech analysis with pitch extraction by using an autoregressive analysis and a downsampling procedure. It is fast but not very accurate and bases itself on a voice/voiceless pre-segmenting. It works within the range 60÷400 Hz extracts pitch from a speech file mainly by testing the energy of the signal against an energy threshold.

The algorithm developed by Reetz (Reetz, 1989) is more accurate and more promising with female voices, i. e. voices with higher frequency values of the pitch. It works on time domain, is resistant against non periodic noise and works within the range 50÷1000 Hz. It can be divided into four major steps: data reduction, logical filtering, chaining and post processing (Reetz, 1989). Data reduction performs a conversion of a speech signal into a set of prominent peaks. Logical filtering (the second step) is composed by four separate tests (noise test, amplitude test, distance variation test, combined distance-amplitude test and distance pattern test). Chaining is the third step and aims at the definition of an optimal pitch track that joins the peaks and, last but not least, post processing is the step that eliminates segments that are too high, low or short.

Both the algorithms are designed to work on input files containing speech and to produce files containing the extracted pitch: Sift produces files of type .LPC whereas Reetz of type .PITCH.

The aforesaid algorithms have been implemented within Audio whereas Pit has been designed to access the file formats produced by Audio (and therefore .LPC and .PITCH).

We note here that Audio, (Foti, 1988), on its own, can even display the content of one speech file and that of the corresponding pitch file on a single window.

What we needed was an application oriented towards the study of F0 and thus we designed and developed Pit.

The design was carried out so to have Pit either as a stand-alone application or as a command of Audio. The latter possibility allows the establishing of a pipeline among Audio and Pit: in such a pipeline Audio works as a producer and Pit as a consumer.

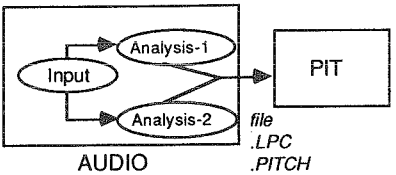

Figure 5. AUDIO and PIT.

Audio, therefore, produces pitch files for Pit that displays the result of such analysis. The same holds for any other application provided that it produces pitch files with a compatible format.

Anyway, Pit is an application oriented towards visualisation so that, even if it is inserted into a graph or a pipeline of applications, it represent a terminal module. Its nature, therefore, prevents Pit from producing new pitch files so to maintain full consistency between a speech file and the corresponding pitch file.

CONCLUSIVE COMMENTS AND FUTURE PLANS

At present Pit should be considered as a beta-release because it is still in the validation phase and requires a, hopefully short, debugging period. As to its future, we are planning both its porting on a more progressive platform such as a DEC Alpha and some improvements. Such improvements involve essentially the procedure of access to the pitch files and the possibility for a user to examine more easily the pitch files when they are displayed all together on the visualisation window.

NOTES

(1) In this paper we use the terms application and program as synonyms.

(2) In this paper we use the terms pitch and F0 with the same meaning though we prefer F0 to denote the graph and pitch as a reference to files. We note moreover that within the figures, distinct graphs are represented with lines of different thickness and feature.

(3) *resize* is a shortcut for the phrase *change the size of* whereas *resizing* is *the action of changing the size of*.

(4) We note that, whenever they are present, these buttons retain the same meaning, with the exception of *delete* that generally applies only to the window on which it appears and only during the starting phase assumes this meaning.

(5) In the figures of the present paper, for obvious graphic reasons, we have used lines of different styles instead of names and colours.

(6) With the term phrase we denote the content of a single pitch file whereas with the term word we denote a segment of the same file. We trust in the benevolence of the purists.

REFERENCES

Cioni, L. (1993) *Pit: un semplice programma per il confronto delle F0*, Atti delle IV Giornate di studio del G.F.S., Torino, 11-12 Novembre 1993.

Cioni, L. (1994) *Integrazione di applicazioni per l'analisi del segnale vocale*, Atti del XXII Convegno dell'Associazione Italiana di Acustica, Lecce, 13-15 Aprile 1994.

Foti, E. (1988) *Audio source code*, CSELT, Torino.

Reetz, H. (1989) *A fast expert algorithm for pitch extraction* , Proceedings of the European Conference on Speech Technology, Paris, 1989.