16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

EXTENSION OF EASYPAS SOFTWARE FOR THE LEARNING OF IMAGE AND AUDIO DIGITAL PROCESSING

Begoña García Zapirain, Ibon Ruiz Oleagordia , Amaia Méndez Zorrilla, Javier Vicente

University of Deusto
Avda. Universidades, 24. 48009, Bilbao
Spain
mbgarcia@eside.deusto.es , ibruiz@eside.deusto.es, amendez@eside.deusto.es, jvicente@eside.deusto.es

ABSTRACT

This article describes the extension of easyPAS software for audio and image. EasyPAS is an educational tool in the scope of digital image processing and audio signals for both undergraduates and postgraduates. The extension is intended to be an independent learning system where the student can carry out the mathematical and graphical study in Octave/Matlab for the analysis and processing, of musical effectsfor example, as well as detections of contour or calculation of negatives on images. In addition, from the examples contributed to the application, where the characteristic parameters of each system will be varied to evaluate their effect, the student will be able to develop new plugins with problems and a new simulated audio and image system in Octave/Matlab. These plugins will be in XML or Java format.

1. INTRODUCTION

In the learning process of any discipline, experts in pedagogy and teaching experience itself have proved that students assimilate knowledge better if they are given visual prompts they can interact with. Bearing this in mind, work began on developing interactive examples in which the different areas of digital processing could be analysed.

In order to achieve this, it was decided to develop a free software application with a Java user environment which included the basic signal processing systems plugins implemented in Octave/Matlab, such as analogical modulations, digital modulations, filters, spectral analysis and digital processing of signals and systems in discrete time.

Having checked and verified the success of the above application in terms of quality and speed in the learning process of students, it was decided to incorporate the examples of audio and image digital signal processing into this tool. In the case of audio signals, the plugins included refer to the application of musical effects, such as reverberation, echo, delay, etc. When working with images, such common and useful treatments as the detection of contours or calculation of negatives could be applied. However, two aspects are fundamental in the learning process; on the one hand, students should verify the importance of each parameter not only by carrying out the mathematical problems on paper, but also by making both a graphic and time-spectrum analysis using software tools. On the other hand, future engineers need to be able to carry out unaided laboratory practices and projects that they themselves will have to design, program and simulate the signal processing algorithms. This is all done in Octave [3], but with the inconvenience of not being able to produce graphic user interfaces.

Therefore, so that the students may equip their Octave algorithms with a user interface, it was necessary to develop the joPAS API [10]. This API enables the rapid implementation of the GUI in Java, and maintains the signal processing calculations in Octave.

Secondly, an application was developed that allows GUIs of a predefined structure to be equipped with Octave functions, simply by defining a straightforward XML file in which the elements required by the user interface are specified. Then the possibility of creating plugins in Java was added. These needed to have a defined structure so that the application could detect them and add them to their menu.

Within this context, the need arose to add a range of more advanced and specific processing to the area of audio and image signals, which is what is presented in this paper.

2. OBJECTIVES

So as to focus completely on the application presented in this article, the aims proposed at the beginning of the easyPAS development project, as well as during the extension of audio and image processing systems, are listed below. As a starting point, the general objective was to develop a free-software learning system with theoretical and practical signal-processing contents by means of Octave simulation in a Java graphic environment. In order to achieve this, it would be necessary to fulfil a number of specific aims that could be organised into two groups: learning and technological objectives.

The learning objectives set out in this project are as follows:

- To enable a computerised verification of results by means of interaction between the student and the application in each one of the examples.

Therefore, the student would be able to check all the time-spectrum characteristics of the signals (audio and images) or the system being studied (musical effects, contour detection, etc). Examples that have been solved on paper in class will be included, and students will be able to check the results of these examples on a PC.

- To offer the possibility of **creating** new plugins related not only to problems proposed to undergraduates by the professor in the learning process, but also the design and simulation of more advanced algorithms corresponding to a project at postgraduate level. In this case, the students must both provide the theoretical solution and carry out the whole development, simulation and implementation of the system themselves.

All this takes two quite different student profiles into consideration. On the one hand, engineering students, who only wish to be able to understand digital signal processing systems at user level (as could be the case of computer engineers). On the other hand, we have those students who should not only understand but also be able to design, simulate and implement new digital signal processing algorithms
As regards the technological aims, the characteristics that the software should meet are as follows: it must be based on free software, be multi-platform, intuitive, complete, open coded, scalable and updateable.

3. METHODS

At this point, the methods used for the development of the easyPAS tool will be described. The different existing alternatives are set out in Table 1, as well as the authors' final choice.

	Choice	**Alternative**
Programming language	Java	C/C++
Scientific language	Octave	Matlab
Graphic interface	joPAS/XML	Octave-GTK

Table 1 - Comparison of technologies

The main characteristics of the technologies employed are set out below, as well as the justification for their choice as regards the various alternatives available.

3.1 Octave

Octave is a high-level language for numerical calculation, whose syntax is compatible with MATLAB and is developed by the free software community [5][6]. Being an Open Source project, Octave does not have the same limitation as Matlab, that is, it is not necessary to pay for a licence in order to be able to execute applications developed through this environment.
Octave is particularly oriented towards the scientific world. Among its main differences from other programming languages, the following stand out:

- Native matrix operation.
- Native operation with complex numbers.
- Language is interpreted.

These characteristics mean that scientific algorithms can be developed in a far shorter time then in other programming languages. Therefore, Octave is the ideal language for the development of digital signal processing algorithms, digital image processing, control systems, statistics,etc.
Furthermore, there a great many toolboxes that allow the user to avoid having to start from scratch when wishing to deal with a particular subject matter.

3.2 joPAS

The joPAS API [7], developed by the PAS group at the University of Deusto, permits the user to use the calculation power of Octave from a Java application.
There is a project called Octave-GTK [8] which provides a number of characteristics similar to the joPAS API, but this did not adjust suitably to the established aims.
The main difference between the projects is the language with which the Octave wrapper is implemented; in Octave-GTK C is used, whereas in joPAS the language used is Java, the programming language our students are accustomed to working in.
With joPAS the user can program applications in Java, assigning all the mathematical calculations in Octave. joPAS is thus used as a uniting link between Java and Octave. Using joPAS means that variables in Java can be converted into Octave variables, Octave functions can be executed and variables in Octave can be converted into Java.

3.3 XML

XML is an extensible brand language developed by the World Wide Web Consortium (W3C). It is a simplification and adaptation of SGML and enables the grammar of specific languages to be defined. Although it is being used mainly for internet purposes, XML is proposed as standard for the exchange of structured information between various platforms.
The main advantage of using XML files is the easiness with which these files can be read and edited. Any user can edit an XML file from any text processor that has the capacity to produce files in plain text. XML is a very simple language to read, interpret and modify.
XML technology has led to the creation of a straightforward language for the configuration of processing. The user defines the entry parameters, the graphics that the application should access, as well as the functions the application should use in order to process the entry signal.

3.4 Reverberation

A set of plugins corresponding to different musical effects has been added so that it can be applied to an audio signal and image processing, such as the detection of contours or calculation of negative of the images. As it is impossible to explain them all due to space restrictions on this paper, the authors have focused on one of them as a practical

representative example of the set being presented. Therefore, we will concentrate on the "Reverberation" musical effect, whose characteristics are describes below.
As an introduction, it is necessary to point out that the reverberation effect is a phenomenon derived from the reflection of sound, consisting of a slight prolongation of a sound once the original has faded away, due to the reflected waves. The latter undergo a delay of less than 100 milliseconds, which is the value of acoustic persistence, the time that corresponds to the distance of 34 metres covered at the speed of sound. When the delay is greater, we are no longer dealing with reverberation but echo.
In a small enclosed space the reverberation might turn out to be unappreciable, but the bigger the space, the better the ear perceives this delay or slight prolongation of the sound. So as to determine what the reverberation is like in a particular space, a series of physical parameters is used, one of which is known as reverberation time.

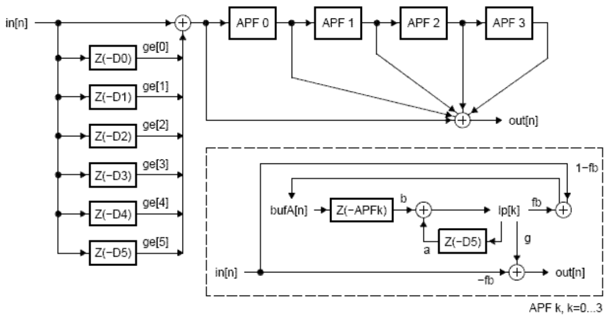


Figure 1 - Block diagram of reverberation effect

This reverberation effect can be defined with the block diagram in Figure 1.

This phenomenon can be generated both analogically and digitally for application to audio signals. The most common parameters will be the following: the times of delay and sound length, and the intensity level of the replica.

4. DESIGN

In this section, what follows is a description of how new audio and image processing extensions based on the abovementioned methods can be designed, so that the easyPAS application can recognise and incorporate them.
In order to this, this paper focuses on a particular example: the musical effect of "reverberation", although the cases of the musical effects "delay", "tremolo", "pitch shifter" or "distorsion" have also been dealt with. The way to implement it is by the definition of an XML-Plugin, which allows exercises to be carried out swiftly and simply through an XML-configured file. The limitation of this method is that the interface generated by the application has a predefined and closed structure, so the user can only specify the number of components that will be visible in each area.

4.1 XML-Plugin for "Reverberation"

In this type of plugins both the graphic interface and the simulation algorithm are defined. Therefore, by using the XML, the contents housed in each one of the interface's six areas is defined. The latter six areas the reverberation example interface has been divided into are the following:

- Plugin title: "Reverberation"
- Graphic representation area.
- Selection of graphic group to be visualised.
- Entry parameter area.
- Processing execution buttons.
- Text area for theoretical description of algorithm.

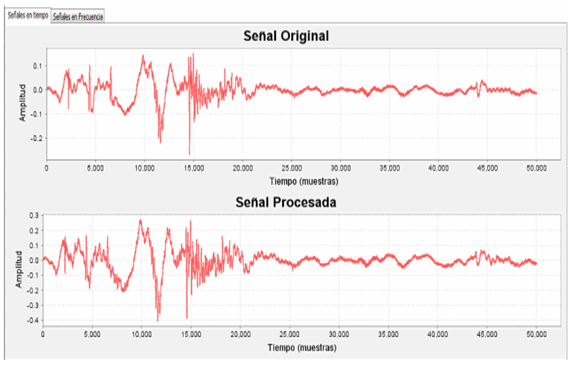


Figure 2 - Diagram of configuration file's definition methodology

The reverberation plugin basically comprises three parts. The first is the description zone, containing the diagram of the system blocks and a photo of a pedal of this effect for guitar. The second is the graphic zone, which is made up of two tabs. In one of them, the entry and exit signals in the time domain are represented, and in the other, the same signals, but in the frequency domain.

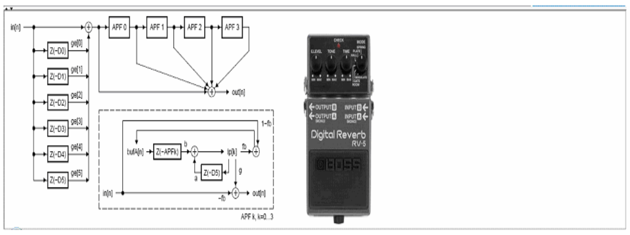
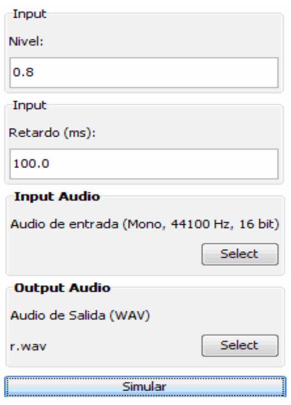


Figure 3 - Diagram of configuration file's definition methodology

Finally, we have the panel where the effect's parameters are introduced, as well as the field for selecting the .way file, and also the name that is to be saved.



Input
Nivel:
0.8
Input
Retardo (ms):
100.0
Input Audio
Audio de entrada (Mono, 44100 Hz, 16 bit)

Output Audio
Audio de Salida (WAV)
r.wav

Figure. 4 - data window for reverberation effect plugin

In order to define the graphic interface, a specific XML language was identified. The defined labels serve both to indicate which elements will appear in each area and to specify the algorithm to be executed.
The labels defined in this language are: “Category”, “Title”, “Description”, “Input”, “Button” and “Function”.

In order to generate the plugin configuration XML file, the desired elements for each area must be identified. Once this has been done, the description of the XMLfile contents can be started. This description of elements could be grouped together in accordance with the previously mentioned six areas.

4.2 joPAS-Plugin

The implementation of plugins using the joPAS API is much more powerful, particularly in the creation of graphic interfaces. In this case, the user replaces the creation of an XML-configured file with the creation of a .java file. The users can thus freely develop their creativity, making more complex and powerful plugins than in the previous case, as they are not restricted to a predefined structure. Therefore, the user will be able to use databases, files and communication via internet or any other resources available for Java language.
Even so, the user does have a few simple restrictions:

- The Java class of the plugin must correspond to the jopasPlugin class.
- The class constructor must receive a reference and an instance of the Jopas class and a reference of the JFrame-type class as entry parameters.
- The getCategory and getTitle methods must be implemented.

Furthermore, the class constructor must also receive a reference to the joPAS API in execution and a reference to the JFrame in which the plugin will be visualised as parameters, because the plugin implemented by the user is a panel visualised within easySP. Since easySP uses the joPAS API to make calls to Octave and to execute the signal processing functions implemented by the user, it is recommended that only one instance of the joPAS class is created. This is necessary in order to optimise the application’s use of resources as it forces all the plugins, whether those in XML files or joPAS-Plugins, to use the same joPAS instance. The XML-Plugin user is unaware of the use of joPAS, as he/she does not codify any Java lines. Nevertheless, the joPAS-Plugin users must use the abovementioned API.

5. RESULTS

As a consequence of executing the specified aims, the Audio/Image-easyPAS tool has been obtained, a tool which covers learning how to process both audio and image signals from the two proposed depth levels. With this extension, easyPAS continues to be a truly simple and comfortable application to use at user level. Once the student chooses the module they wish to practise with, they visualise a series of windows like those that can be seen in Figure 5.
The example in Figure 5 corresponds to a simulation of the reverberation effect in which the student can modify the basic parameters defining the characteristics of the effect and can also analyse its behaviour according to the variation of these parameters. At the bottom of the window, the student finds a theoretical explanation of the system’s behaviour, which can be verified by carrying out various trials.

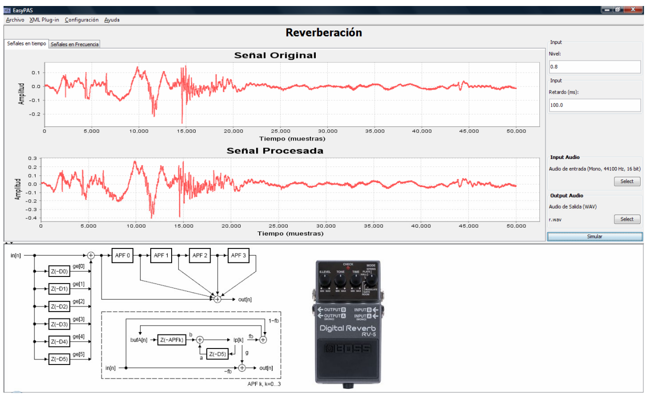


Figure 5 - Diagram of configuration file’s definition methodology

At the second depth level, students can develop new modules by following the defined XML structure, or can analyse the already existing ones, in order to check which Octave sentences are necessary to implement the module proposed.

Along with the functional results, it is necessary to assess the efficiency of our students’ learning how to use the tool developed. This was possible thanks to an anonymous opinion poll -whose items can be seen in Table II- carried out on a control group of ten scholarship holders from the telecommunications laboratory doing a degree in Telecommunications Engineering at the University of Deusto. The 10 subjects of the survey had previously taken the subject of Digital Signal Treatment in its traditional form.

ITEM	SCORE
Is the application intuitive?	4.1
Is it easy to use?	3.5
Do the application's plugins help in understanding	4.5
Is a lot of time needed to master the application?	2
Complexity of XML-plugin design	3
Functionality that can be implemented with XML-	3.5
Complexity of joPAS-plugin design	4.5
Functionality that can be implemented with joPAS-	4
Does the design enable going deeply into the contents	4.5
Do you find it more motivating designing plugins	4.5
General degree of satisfaction with the application	4

Table 2 - Satisfaction survey results

They therefore had all the necessary knowledge and could also give a valid opinion on the tool's use.

CONCLUSIONS

The application developed has hugely improved the work of both the professor and the student, allowing not only the simulation of contents but also enhancing the student's creativity. Professors have at their disposal a versatile tool that enables them to transmit knowledge about digital signal processing in the scope of both audio and images, by implementing small plugins as a back-up to the theoretical contents they wish to teach. Students can better assimilate the knowledge of audio and image processing thanks to a tool they can interact with. In addition, they can develop new modules without needing to know any other programming languages, except Matlab/Octave.
It is necessary to highlight the fact that the professors experiencing the use of Audio/Image-easyPAS noticed an increase in student motivation as regards the degree of interest shown in other years when the traditional method was used.
A wizard to generate the contents of the XML is currently being worked on. This will enable students to avoid having to specify the configuration file, so that they can devote all their efforts to implementing the algorithm in Octave.
In addition, this allows the creation of joPAS-Plugins, which means that users with knowledge of Java can create much more powerful plugins with new systems (filters, modulators, etc.).
As a final conclusion, it should be pointed out this API's result has been very satisfactory for both developers and users, as it provides a very powerful tool for the rapid development of applications for audio/image signal processing. It also has the advantage of being able to be applied in the context of Biomedical Engineering research, specifically in oesophageal speech processing projects.

ACKNOWLEDGEMENTS

The authors of this article would like thank the students taking part in the Project for informing us of their experiences with the application, as well as for actively contributing to its extension by producing new plugins.
This research was partially carried out under grant TEC2006-12887-C02-02 from the Ministry of Science and Technology of Spain.
Authors also wish to thank the support of INRIA's EUROMED 2007 project, and University of Deusto´s REVOES project.

REFERENCES

[1] Begoña García, Javier Vicente, "*Herramienta para la Simulación e Implementación Real de Sistemas Discretos FIR e IIR*" in Proc. TAEE'02, Las Palmas, Spain, 2002.
[2] J, Angulo, B. García, J, Vicente, I. Angulo, "*Microcontroladores avanzados dsPIC*", International Thomson Editores, 2005.
[3] M Murphy, "*Octave: A Free, High-Level Language for Mathematics*" in Linux Journal, 1997.
[4] WS Gan, YK Chong, W Gong, WT Tan, "*Rapid prototyping system for teaching real-time digital signalprocessing*" in IEEE Transactions on Education, 2000.
[5] Kurt Hornik, Friedrich Leisch, Achim Zeileis, "*Ten Years of Octave Recent Developments and Plans for the Future*" in Proc. DSC 2003, Vienna, Austria, 2004.
[6] JW Eaton, "*Octave: Past, Present, and Future*" in DSC 2001, Vienna, Austria, 2001.
[7] Javier Vicente, Begoña García, Amaia Mendez, Ibon Ruiz, Oscar Lage, "*Teaching Signal Processing Applications With joPAS: Java To Octave Bridge*" in Proc. EUSIPCO 2006, Firenze, Italy, 2006.
[8] Muthiah Annamalai1,Hemant Kumar, Leela Velusamy, "*Octave-GTK: A GTK binding for GNU Octave*", The Cornell University Library, 2006.
[9] J Campbell, F Murtagh, M Köküer, "*DataLab-J: A Signal and Image Processing Laboratory for Teaching and Research*", in IEEE Transactions on Education, 2001.
[10] WS Gan, "*Teaching and Learning the Hows and Whys of Real-Time Digital Signal Processing*" in IEEE Transactions on Education, 2002.