On the Use of the 'Pure Data' Programming Language for Teaching and Public Outreach in Speech Processing

Roger K. Moore

Speech and Hearing Research Group, University of Sheffield, UK

r.k.moore@dcs.shef.ac.uk

Abstract

Many educational institutions include a course on speech processing as part of their regular curriculum. Such courses usually cover basic principles in acoustics, phonetics and speech signal processing, and they often incorporate practical lab classes using specialised speech processing toolboxes such as 'Praat'. Such toolkits are valuable resources for teaching and learning, but they often involve scripting solutions based on prescribed templates and non-real-time processing, and they do not lend themselves to be used by non-specialists, such as members of the general public or schoolchildren. This paper introduces the Pure Data open-source real-time visual programming language, and presents examples of its use for teaching and public outreach in speech processing. The 'Show & Tell' session will present 'live' examples as well as hands-on interactive demonstrations to illustrate its value.

Index Terms: speech processing, pure data, real-time, teaching, public outreach

1. Introduction

Many educational institutions around the world include a course on speech processing as part of their undergraduate/postgraduate curriculum. Whether it is within a Phonetics, Engineering or Computer Science Department, such courses usually cover basic principles in acoustics, phonetics and speech signal processing, and they often incorporate practical lab classes using specialised speech processing toolboxes (e.g. Praat [1], VOICEBOX [2] or SFS [3]), often in conjunction with a standard scientific programming environment such as MATLAB®.

Such toolkits are valuable resources for teaching and learning, but they often involve scripting solutions based on prescribed templates and non-real-time processing. The latter is particularly restrictive when it comes to explaining a particular algorithm's dynamic response to an incoming signal or for generating 'live' output that can be controlled 'on the fly'. Such toolboxes are also highly technical, and they do not lend themselves to be used by non-specialists, such as members of the general public or schoolchildren.

For these reasons, in 2009 the long established Speech Processing course in the Computer Science Department at the University of Sheffield was redesigned to incorporate examples, live demonstrations, lab classes and programming assignments using the Pure Data programming language [4]. Since then, over 350 students (at 3rd-year undergraduate and Master's level) have taken the course, and have provided positive feedback on the new more interactive approach. In addition, some of the code was re-purposed to create hands-on exhibits for a number of public outreach events (organised by CREST – the 'Creative Speech Technology Network' [5] funded by the UK Engineering and Physical Sciences Research Council.

This paper introduces the main features of the Pure Data language, highlights its advantages for real-time speech processing and presents examples of its use in teaching and public outreach. A number of real-time examples will be shown 'live' during the 'Show & Tell' session, including hands-on interactive demonstrations.

2. Pure Data (Pd)

Pure Data - known as 'Pd' - is an open-source real-time visual programming language that is designed to operate on audio, graphical and video signals [4]. Originally authored by Miller Puckette at IRCAM in Paris, Pd is a free alternative to Max™ – a programming language popular in the professional music industry [6]. Pd is available for Windows, Mac OSX and GNU/Linux platforms, and Pd-extended is the recommended version to download.

Pd is an object-oriented dataflow programming language in which functions are created graphically and which run immediately they are instantiated. A Pd program - known as a 'patch' - consists of objects, connections and data. Objects are functions such as '`print`', '`+`', '`min`' etc. and they connect with other objects via inlets and outlets. Connections between objects carry data; thin connections carry messages, and thick connections carry audio. Pd also provides various GUI objects, such as sliders, graphs and buttons. Figure 1 illustrates a Pd patch for real-time simulation of the larynx.

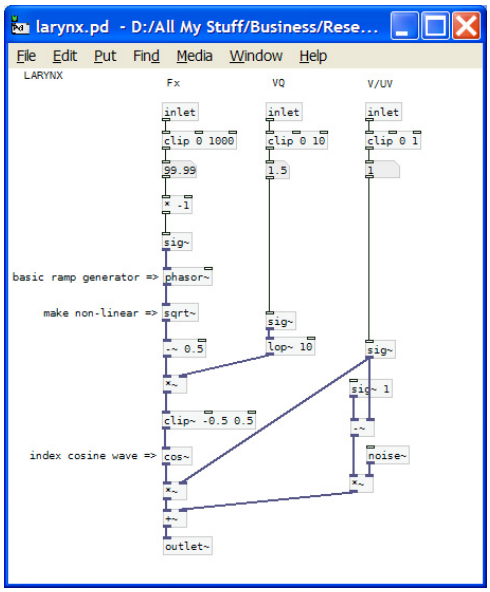

Figure 1: *Example of a Pd 'patch' that simulates the larynx in real-time. The three inlets take values from a GUI patch for fundamental frequency (Fx), voice quality (VQ) and degree of voicing (V/UV). The output is a 'live' audio signal.*

One of the main reasons to choose Pd as a programming language is that it makes it extremely easy to deal with real-time audio. Not only can audio be input and processed in real-time, but audio can also be generated and output in real-time – all at the same time. It is for this reason that Pd is very often used to interface with MIDI devices and to synthesise electronic music. Andy Farnell's textbook - *Designing Sound* - is an excellent resource that covers a wide range of Pd-based applications in what he calls 'procedural audio' (especially useful for generating rich soundscapes in computer games) [7].

As well as real-time processing, Pd objects are pre-compiled. Hence a Pd program is actually running *while it is being edited*. This means that, not only is debugging particularly straightforward, but it is also possible to see/hear the consequences of the connections between the different parts of a program. For example, the necessity of both the real and the imaginary parts of an FFT can be demonstrated 'live' by disconnecting one of them to hear the resulting distortion – all with two clicks of the mouse. Also, since the underlying codebase is C, it is easily possible to write your own Pd objects. For example, a student at Sheffield has produced a Pd-based real-time LPC analyser and synthesiser, and the author of this paper has written a Pd object for an acoustic waveguide (to form the basis of an articulatory synthesiser).

3. Pd in Teaching

The Sheffield course on Speech Processing consists of a series of 20 one-hour lectures coupled with 20 hours of lab classes. Assessment takes place through two on-line quizzes and two lab-based programming assignments. The course is effectively in two halves: the first part covers the principles of speaking and hearing (including acoustic/articulatory-phonetics, phonology and prosody) and the second part covers algorithms for speech processing (including digital signal processing, frequency analysis, the Fourier and Z transforms, filtering, linear prediction and cepstral processing).

A total of 54 Pd-based examples have been constructed for the course, all of which are demonstrated live during lectures to illustrate particular speech processing techniques. The examples range from simple demonstrations of resonances in acoustic tubes, to simulations of the cochlea, formant synthesis, channel vocoding, speech detection, and homomorphic filtering – all demonstrated in real-time. All programs are made available to the students for personal study and for use in their assignments, together with various Pd-based tools (such as a real-time display – see Figure 2).

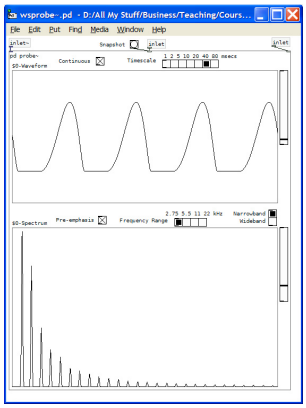

Figure 2: *Pd-based real-time waveform/spectrum display (connected to the output of the patch shown in Figure 1).*

4. Pd in Public Outreach

Since Pd makes it very easy to construct real-time interactive speech processing, it has been used very successfully to produce hands-on exhibits at public events as well as to create simple educational activities for schoolchildren. For example, an exercise known as 'MakeSpeech' was produced for use by 15 year-old potential computer science students in which they were introduced to the human voice by leading them through a step-by-step sequence culminating in a hand-controlled formant synthesiser. Prizes were offered for the student who could produce the best output speech.

As an example of engaging with the general public, a Pd-based exhibit entitled *Prof. Moore's Digital Voice Factory* allowed users (especially young children) to experiment with their voices– see Figure 3. The main technique underlying the Digital Voice Factory was phase vocoding, and the exhibit allowed speech to be reversed and modified (e.g. to sound robotic). The GUI was programmed in GrIPD in order to avoid users inadvertently editing the code, and the whole thing was designed to be operated using a touch screen.

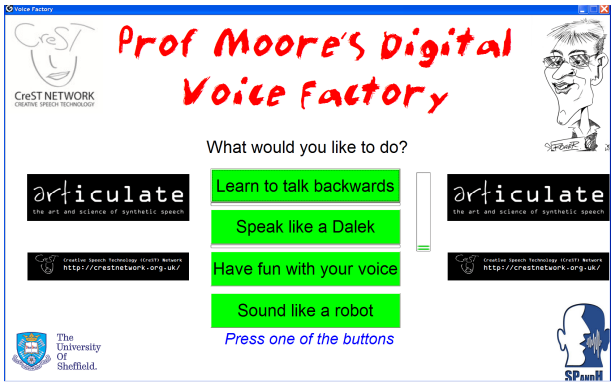

Figure 3: *Screenshot of the Pd-based 'Digital Voice Factory' which allows users (especially children) to experiment with their own voices.*

5. Conclusion

This paper has introduced the Pure Data programming language as a powerful open-source tool for real-time speech processing. Examples have been given of its use in teaching and public outreach, and its capabilities will be demonstrated live during the hands-on interactive 'Show & Tell' session. Finally, it may be of interest to note that Pd also provides an excellent rapid prototyping environment for developing all kinds of real-time systems (including sensorimotor control for robots).

6. References

[1] Praat: doing phonetics by computer, http://www.fon.hum.uva.nl/praat/

[2] VOICEBOX: Speech Processing Toolbox for MATLAB, http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html

[3] Speech Filing System, http://www.phon.ucl.ac.uk/resource/sfs/

[4] Pure Data portal, http://puredata.info/

[5] The Creative Speech Technology Network (CreST), http://crestnetwork.org.uk/

[6] Max, http://cycling74.com/products/max/

[7] Farnell, A., *Designing Sound*. London: Applied Scientific Press Ltd, 2008.