Python For Audio Signal Processing

Linux Audio Conference 2011
National University of Ireland, Maynooth
7th May 2011

John Glover
The Sound and Digital Music Research Group
National University of Ireland, Maynooth

Overview

Look at Python programming language and libraries for scientific computing:

- NumPy
- SciPy
- Matplotlib

Overview

Show how Python was used to create 2 audio programming libraries:

- Modal
- Simpl

Describe Python integration with other tools:

- The SndObj Library
- Pure Data

Introduction

There are many common problems in audio signal processing:

- loading sound files
- communication with sound cards
- DSP tasks (filtering, Fourier analysis, etc)

Generally makes sense to rely on frameworks/libraries for these sort of tasks, particularly when prototyping.

Introduction

Audio programming libraries exist for many general purpose programming languages, eg: C/C++

Can result in long development times

This is one of the factors that fuels the popularity of tools such as Matlab.

Introduction

Problems with MATLAB:

- Expensive
- Not open source, harder to share work

GNU Octave: an open source alternative to MATLAB

- Interpreted
- Similar syntax to MATLAB

Introduction

Other issues with MATLAB/Octave:

- Complex development greatly aided by tools such as IDEs, debuggers, profilers. Exist in some form for MATLAB/Octave, more time spent on tools for a domain with limited scope.
- Don't integrate as well with compositional tools such as Csound and Pure Data, or with web frameworks, mobile applications, etc.

Introduction

Desirable features in an environment for prototyping audio signal processing applications:

- Power/flexibility of an open source, widely adopted general purpose programming language
- Quick development process of interpreted, dynamic languages
- Large collection of signal processing tools/libraries
- Easy integration with existing tools/solutions

Python meets all of these criteria.

Python

- Open source
- Runs on all major platforms
- Widely used
- Actively developed
- Vast array of libraries and development tools

Python

- Multi-paradigm
- Clean syntax, very readable code (executable pseudocode)
- Interactive interpreter, useful for experimentation / live coding
- Very easy to extend using C/C++, so can optimise later.

Python

- Can be embedded into existing applications
- Documentation can be generated automatically from comments and source code
- Python bindings exist for cross-platform GUI toolkits (eg: Qt)

More information and comprehensive tutorial at: http://python.org

Python For Scientific Computing

Python's scientific computing prowess comes largely from a combination of 3 related modules:

- NumPy
- SciPy
- Matplotlib

Python For Scientific Computing

NumPy:

- Adds homogeneous, multi-dimensional array objects and functions that perform efficient calculations using them.
- Syntax is like a Pythonic MATLAB
- Written in C, easily extended via C API
- Includes f2py to create Python extension modules from Fortran code

Python For Scientific Computing

SciPy:

- Builds on top of NumPy, providing modules dedicated to common issues in scientific computing
- Similar to MATLAB toolboxes

A full list of modules is available at: http://docs.scipy.org, but examples include:

- scipy.io: file input/output
- scipy.fftpack: Fourier transforms
- scipy.signal: signal processing functions
- scipy.interpolate: linear interpolation, cubic splines
- scipy.linalg: linear algebra

Python For Scientific Computing

Matplotlib:

- Library of 2D plotting functions
- Adds ability to visualise data from NumPy arrays
- Produces publication ready figures in a variety of formats
- Can be used interactively or from scripts, with GUI toolkits or from web apps.
- Similar functionality to GNU plot

SciPy Examples

- Plot audio waveform
- Plot FFT data

Simpl

Simpl is an open source library (GPL) for sinusoidal modelling written in C/C++ and Python.

The aim of this project is to tie together many of the existing sinusoidal modelling implementations into a single unified system with a consistent API, as well as provide implementations of some recently published sinusoidal modelling algorithms.

Simpl is primarily intended as a tool for other researchers in the field, allowing them to easily combine, compare and contrast many of the published analysis/synthesis algorithms.

Simpl

Simpl breaks the sinusoidal modelling process down into three distinct steps:

- Peak detection
- Partial tracking
- Sound synthesis

Implementations have a Python module associated with every step which returns data in the same format, irrespective of its underlying implementation.

This allows analysis/synthesis networks to be created in which the algorithm that is used for a particular step can be changed without effecting the rest of the network.

Simpl: Use of SciPy

All audio in Simpl is stored in NumPy arrays. This means that SciPy functions can be used for:

- Basic tasks such as reading and writing audio files
- Performing additional processing / analysis
- Data visualisation

Simpl: Use of SciPy

Audio samples are passed into a PeakDetection object for analysis, with detected peaks being returned as NumPy arrays that are used to build a list of Peak objects.

Peaks are then passed to PartialTracking objects, which return partials that can be transferred to Synthesis objects to create a NumPy array of synthesised audio samples.

Simpl also includes a module with plotting functions that use Matplotlib to plot analysis data from the peak detection and partial tracking analysis phases.

Modal

Modal is a new open source library (GPL) for musical onset detection, written in C++ and Python.

Consists of two main components: a code library and a database of audio samples.

Modal: Code Library

- Includes implementations of three widely used onset detection algorithms from the literature and four novel onset detection systems created by the authors.
- The onset detection systems can work in a real-time streaming situation as well as in non-real-time.
- Details in our upcoming paper, to be published in the EURASIP Journal on Advances in Signal Processing

Modal: Sample Database

- Database contains a collection of audio samples that have creative commons licensing allowing for free reuse and redistribution, together with hand-annotated onset locations for each sample.
- Includes an application that allows for the labelling of onset locations in audio files, which can then be added to the database.

Modal: Sample Database

- To the best of our knowledge, this is the only freely distributable database of audio samples together with their onset locations that is currently available.
- The Sound Onset Labellizer is a similar reference collection, but was not available at the time of publication.
- The sample set used by the Sound Onset Labellizer also makes use of files from the RWC database, which although publicly available is not free and does not allow free redistribution.

Modal: Use of SciPy

- Extensive use of SciPy, with NumPy arrays being used to contain audio samples and analysis data from multiple stages of the onset detection process including computed onset detection functions, peak picking thresholds and the detected onset locations.
- Matplotlib is used to plot the analysis results.
- All of the onset detection algorithms were written in Python and make use of SciPy's signal processing modules.

Modal: Use of SciPy

- The most computationally expensive part of the onset detection process is the calculation of the onset detection functions, so Modal also includes C++ implementations of all onset detection function modules.
- Made into Python extension modules using SWIG.
- This allows Python to be used in areas that it excels in such as rapid prototyping and in "glueing" related components together, while languages such as C and C++ can be used later in the development cycle to optimise specific modules if necessary.

Python Integration With Other Tools

Going to look at Python integration with:

- The SndObj Library
- Pure Data

However, Python bindings do exist for other popular open-source composition tools:

- Csound
- SuperCollider

The SndObj Library

- Recent version of The SndObj Library comes with support for passing NumPy arrays to and from objects in the library,
- This allows data to be easily exchanged between SndObj and SciPy audio processing networks/functions.

Pure Data

- libpd allows Pure Data to be embedded as a DSP library. and
- It comes with a SWIG wrapper enabling it to be loaded as a Python extension module.
- NumPy arrays can be passed to/from Pd patches.

Conclusions

This paper highlighted just a few of the many features that make Python an excellent choice for developing audio signal processing applications:

- Clean, readable syntax
- Quick development times
- An extensive collection of libraries
- Unrestrictive open source license

Discussed two open source signal processing libraries created by the authors that both make use of Python and SciPy: Simpl and Modal.

Python is easy to extend and integrates well with other programming languages and environments.

Questions?

Software mentioned in this talk:

- Python programming language: http://python.org
- NumPy & SciPy: http://scipy.org
- Matplotlib: http://matplotlib.sourceforge.net
- Simpl: http://simplsound.sourceforge.net
- Modal: http://github.com/johnglover/modal
- The SndObj Library: http://sndobj.sourceforge.net
- libpd: http://gitorious.org/pdlib
- My examples: http://github.com/johnglover/pfasp-examples