Stationary and non-stationary sinusoidal model synthesis with phase vocoder and FFT^{-1}

Clément Cazorla, Vincent Chrun, Bastien Fundaro, Clément Maliet
ENSEEIHT - 3EN TSI 2016-2017

February 10, 2017

Abstract

The present document is to serve as both a technical report and a documentation for the code produced during the long project. As such the first part will explain the theoretical framework and the state-of-the-art of the field. The second part will give some insight about the code structure and the conventions that were adopted and last but not least, the third part will serve as an actual documentation and details every class, methods and attributes.

Contents

Part I

Theoretical framework

1 Sound synthesis

1.1 Frequency domain

1.2 Sinusoidal model

1.3 Phase Vocoder

2 Theoretical synthesis with phase vocoder and FFT^{-1}

2.1 Stationary case

2.2 Non-stationary case

Part II

Code structure and conventions

3 Conventions

In this section we remind the reader of a few coding convention necessary to ensure a seamless work flow and a bug free program as much as possible. Files should contains an entire module (as described in 4.1) not just a single class to limit the number of files and ease the bug tracking. Imports in files should be kept to a minimum and left in namespaces (do not use the **from** module **import** $*$ syntax). It is preferable to import a whole module if more than three elements from the module are needed in the file, otherwise consider the **from** module **import** element1 , element2 syntax to avoid unnecessary memory flooding. If conflicts exists, notwithstanding the number of elements needed, the whole modules are to be loaded with a namespace. Namespaces may be abbreviated to the programmer's convenience however some abbreviation are to be universally respected :

(i) `numpy` should always be imported as `np`

(ii) `matplotlib.pyplot` should always be imported as `plt`

Finally math functions should always come from the numpy module and not python's math module to guarantee a universal behaviour across the program.

3.1 Naming conventions

Naming conventions are freely adapted from Python recommended conventions defined in PEP8 [1], as such :

(i) *Class* should be named in `CapitalizedWord`

(ii) *Methods* and *functions* should be named in `lower_case_with_underscores`

(iii) *Attributes* and *variables* should be names in `lower_case_with_underscores`

(iv) *Instantiation* following the fact that everything is an object in python should be named as *variables*.

Moreover during class declaration, the following principles should be adopted :

(i) Non-public methods and attributes should use one leading underscore.

(ii) Elements that conflicts with python reserved name should use one trailing underscore rather that simplification or a misspelling.

(iii) Accessors or mutators using one leading underscore should be interpreted as properties of their associated attribute. As such it should be guaranteed that they induce a low computational cost.

(iv) Non-public elements that should not be inherited or may cause conflicts during inheritance should use two leading underscore and make use of Python name-mangling.

To seamlessly manipulate both *stationary* and *non stationary* models, class that are inherited in two versions are preceded with either `Stationary` are `NonStationary` respectively.

3.2 Spectrums and sinusoids parameters

4 Code structure

4.1 General structure

4.2 Class structure

Part III

Documentation

5 Core module

5.1 Synthesizer

5.2 StationarySynthesizer

5.3 NonStationarySynthesizer

6 Spectrum generation module

6.1 SpectrumGenerator

6.2 StaionarySpectrumGenerator

6.3 NonStationarySpectrumGenerator

6.4 StationnaryLobe

6.5 NonStationaryLUT

7 Phase Vocoder module

7.1 PhaseVocoder

7.2 StationaryPhaseVocoder

7.3 NonStationaryPhaseVocoder

Conclusion

References

[1] Nick Coghlan Guido van Rossum, Barry Warsaw. Style guide for python code, Aug. 2013.