Long Project with Audiogaming

Additive Synthesis with Inverse Fourier Transform for Non-Stationary Signals

C. Cazorla - V. Chrun - B. Fundaro - C. Maliet

Audiogaming Supervisor :
Chunghsin Yeh

March 03, 2017

INP ENSEEIHT

Content

Part 1

Introduction
The company

AudioGaming
NATURAL BORN INTERACTIVE

- Location: Toulouse, Paris
- Activity: Audio plug-in (VSTs and RTAS)
- Main customers: Film and Video Game Industry (Sony, Ubisoft)
- 10 employees

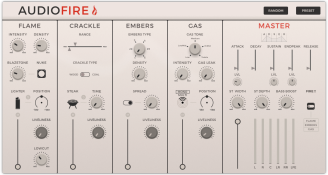


Figure: Audiofire: audio plug-in that recreates fire sound

Introduction
Objective

- We are continuing the Audiogaming long project from 2015 (Emilie Abia, Lili Zheng, Quentin Biache)

Objective : Synthesizing sounds from their spectrum with a FFT^{-1}

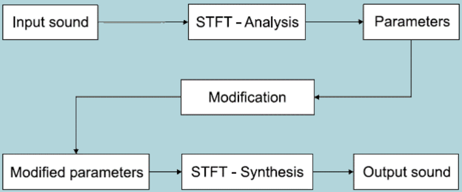


Figure: General approach for modifying a sound in the spectral domain

- We have to implement a new method of additive synthesis $\Rightarrow$ computationally very fast

Introduction

Context of the Project

- 6 weeks only ⇒ Focus on the synthesis method only.

Given codes in Python and Matlab from the 2015 project :

- Python : Analysis estimator of sinus parameters and sinus generation with those parameters (only stationary)
- Matlab : Some reasearch on the Non-stationary synthesis with the LUT of lobes

- We made our own Object Oriented Programmation tree structure in Python
- We remade all the codes to be coherent with the OOP tree structure

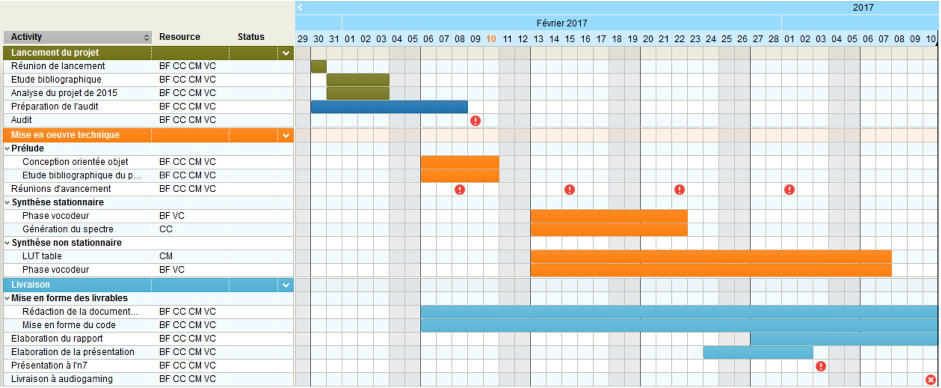
Introduction

Work Environment

Figure: *PyCharm* as Python IDE , *Slack* to communicate,*GitHub* to stock the codes and have a versionning, *Freedcamp* to plan the project events

Introduction

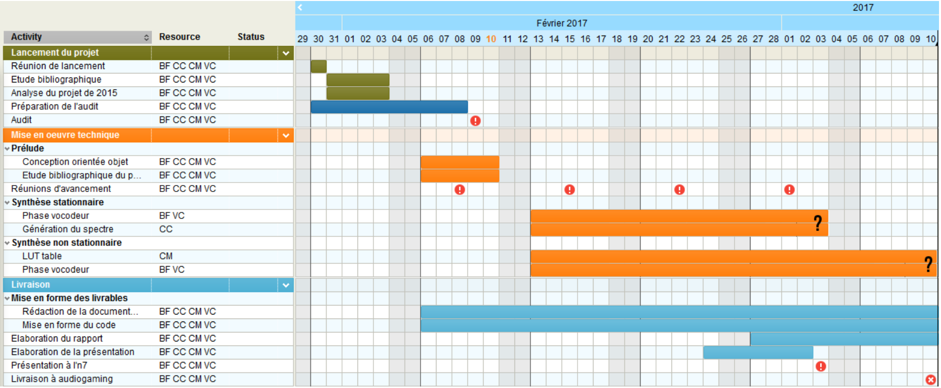
Project Management : Gantt Chart (expected event)



INP ENSEEIHT

Introduction

Project Management : Gantt Chart now



Part 2

INP ENSEEIHT

Additive Synthesis

Time Domain

The sound signal is represented as a sum of N sinusoids:

$$x(t) = \sum_{n=1}^{N} a_n sin(2\pi f_n t + \phi_n)$$

- Very costly to implement
- Impossible to compute in real-time

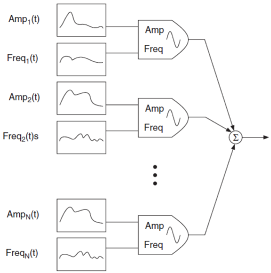


Figure: *The additive synthesis*

Method Overview : Windowing

Analysis

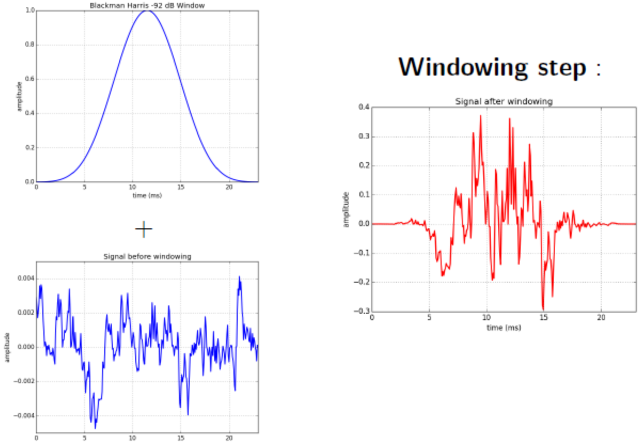


Figure: *Windowing step*

Method Overview : Peak detection in Frequency Domain

Analysis

Peak detection and extraction of parameters by STPT (particular Short Time Fourier Transform):

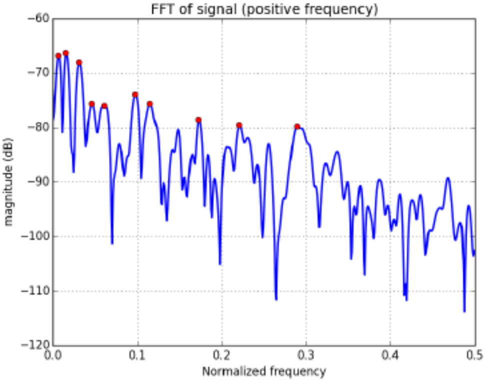


Figure: *Peak detection*

Method Overview : Result (FFT^{-1})

Synthesis

Additive synthesis with FFT^{-1} according to the parameters from the analysis:

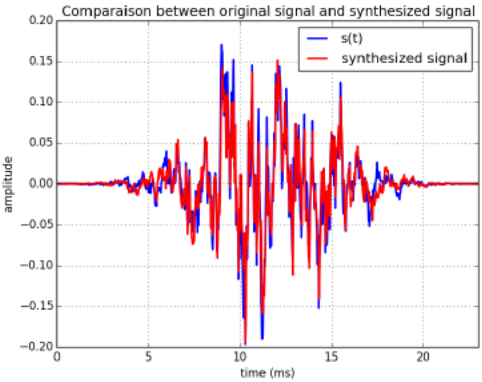


Figure: *Synthesized frame vs Original frame*

Part 3

Stationary Case

Stationary sinusoidal model

Mathematical model :

$$s(t) = a_0 exp[j(2\pi f_0 t + \phi_0)] \tag{1}$$

- 3 parameters: a_0 (amplitude), f_0 (frequency) and ϕ_0 (phase).
- Simplest model but useful for certain kinds of signals.
- Each spectral bin represents a stationary sinusoid.
- $\Rightarrow$ generate a synthetic spectrum with the desired parameters
  $\Rightarrow$ generate a main lobe derived from the Fourier transform of the normalized window w supposedly[1] used during analysis
  $\Rightarrow$ place it at the right position on the spectrum.

[1] Because no actual analysis happened

Stationary Case

Lobe generation

We generate the sinusoids in frequency domain :

- Window the signal to maximize the energy in the main lobe
- We only keep the main lobe for each sine (11 points)
- We assume that the parameters (amplitude, frequency, phase) are already given by the analysis
- We interpolate the relevant bins value if by any chance the wanted frequency $\hat{f}$ is not exactly on a bin, that is to say if $\hat{f} \notin \{\frac{2k\pi}{N}\}_{k=0\ldots N-1}$

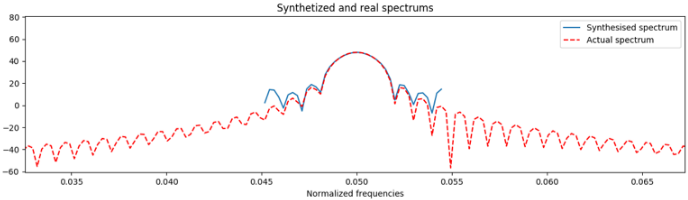


Figure: *Windowed sine lobe*

Stationary Case

Frames separation

The sound signal is a frame-by-frame signal:
The analysis hop size will be called R_a and the synthesis hop size R_s (moving step of the frame)

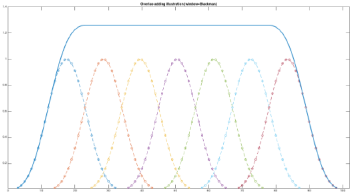


Figure: *Sum of small size Hanning windows*

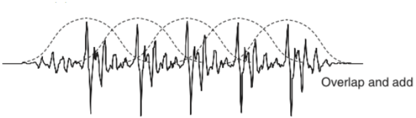


Figure: *Overlap and add*

Stationary Case

Phase Coherence

Phase coherence

The Phase coherence is not a problem in the Stationary case :

- We don't know the window effect on the phase : $f_{w,\hat{f}}(\phi) : \phi \mapsto \tilde{\phi}$
  $\Rightarrow$ We calculate its influence on the first frame and assume the same influence on the other frame.
- We then multiply the generated lobe by $\frac{A}{2}$ and set the lobe phase to $\tilde{\phi} + 2\pi\hat{f}R_a$
- In the purely stationary case, the expected phase shift is the theoretical phase shift :

$$\begin{cases} \tilde{\phi}_i = \tilde{\phi}_{i-1} + 2\pi\hat{f}R_a \\ \tilde{\phi}_0 = \tilde{\phi} \end{cases} \tag{2}$$

Quasi-Stationary Case

What is changing

- In a Quasi-stationary case, the sine wave can change a little bit in frequency. Main problem $\Rightarrow$ Phase coherence !
- We need to implement a a method to correct the phase : Phase Vocoder !

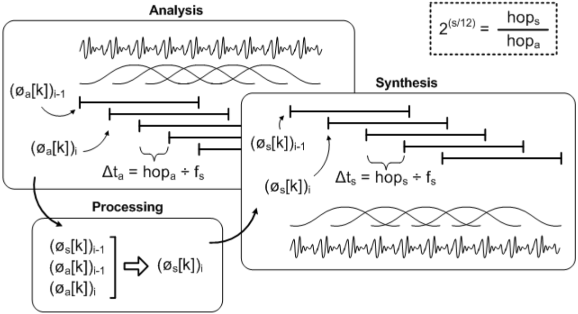


Figure: *Phase Vocoder overview*

Quasi-Stationary Case

What is changing

- In a Quasi-stationary case, the sine wave can change a little bit in frequency. Main problem $\Rightarrow$ Phase coherence !
- We need to implement a a method to correct the phase : Phase Vocoder !

Phase Vocoder

In this case, the phase changing is different from the stationary case. We have to calculate the instantaneous frequency for the kth bin:

$$\hat{\omega}_k(t_a^u) = \Omega_k + \frac{\Delta_p \Phi_k^u}{R_a} \tag{3}$$

Where:

$$\Delta\Phi_k^u = \angle X(t_a^u, \Omega_k) - \angle X(t_a^{u-1}, \Omega_k) - R_a\Omega_k \tag{4}$$

Hence,

$$\angle Y(t_s^u, \Omega_k) = \angle Y(t_s^{u-1}, \Omega_k) + R_s\hat{\omega}_k(t_a^u) \tag{5}$$

We replace the output signal phase by this one.

Non-Stationary Case

A different approach

Mathematical model :

$$s(t) = exp[(\lambda_0 + \mu_0 t) + j(\phi_0 + 2\pi f_0 t + \frac{\psi_0}{2} t^2)] \tag{6}$$

- 5 parameters:
  $(\lambda_0 + \mu_0 t)$ (overall amplitude)
  f_0 (frequency)
  ϕ_0 (phase)
  μ_0 (amplitude change rate (ACR))
  ψ_0 (frequency change rate (FCR))
- The analysis part give us all those parameters
- To manage the influence of the ACR and the FCR on the lobe $\Rightarrow$ Interpolation of Look-up table of already saved lobes with different (ACR,FCR).

Non-Stationary Case

Look up table

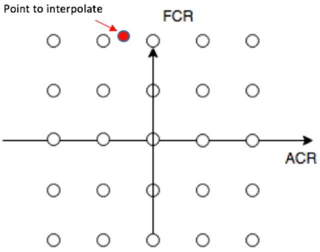


Figure: *ACR/FCR grid*

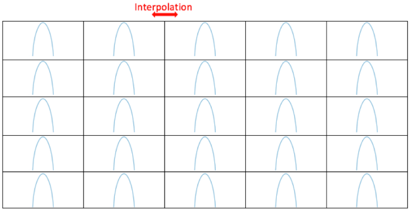


Figure: *Look-up table*

Non-Stationary Case

Phase Vocoder : Scaled-Phase Locking

- We can use the phase vocoder to correct the phase, but the main problem is sine waves that switch from a frequency bin to another bin. The phase might change a lot from one frame to another.
- $\Rightarrow$ We use a refined version of the phase vocoder : Scaled-Phase Locking
  $\Rightarrow$ It takes into account the frequency trajectory of each lobes.

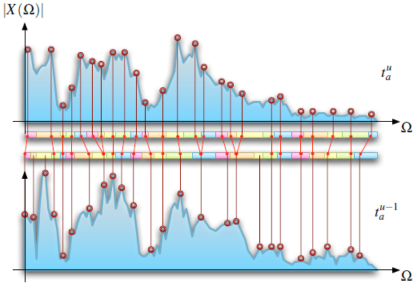


Figure: *Peak coherence from a frame to another*

Non-Stationary Case

Phase Vocoder : Scaled-Phase Locking

Scaled-Phase Locking

We find each region corresponding to each peaks, and then we use the phase vocoder algorithm for each region :
k_0 is the precedent frequency bin for the peak - k_1 is the current frame peak bin :

$$\hat{\omega}_{k_1}(t_a^u) = \Omega_k + \frac{\Delta_p \Phi_k^u}{R_a} \tag{7}$$

Where:

$$\Delta\Phi_{k_1}^u = \angle X(t_a^u, \Omega_{k_1}) - \angle X(t_a^{u-1}, \Omega_{k_0}) - R_a \Omega_{k_1} \tag{8}$$

Hence,

$$\angle Y(t_s^u, \Omega_{k_1}) = \angle Y(t_s^{u-1}, \Omega_{k_0}) + R_s \hat{\omega}_{k_1}(t_a^u) \tag{9}$$

Then for each bin in the region :

$$\angle Y(t_s^u, \Omega_k) = \angle Y(t_s^u, \Omega_{k_1}) + \beta[\angle X(t_a^u, \Omega_k) - \angle X(t_a^u, \Omega_{k_1})] \tag{10}$$

Non-Stationary Case

Scaled-Phase Locking Problem

- Moreover, we do not know for now how to manage the peaks that appear and disappear

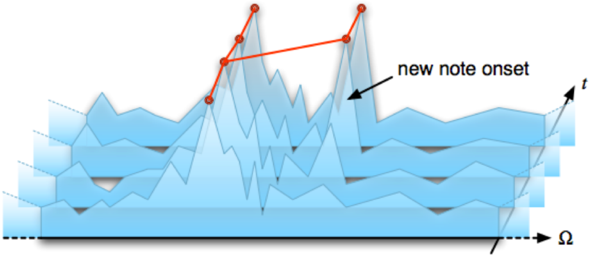


Figure: *Phase-locking problem*

Part 4

Stationary Case

Protocol

- A one second triangular signal consisting in 84 frames
- We first vary the number of harmonics and compare the time of execution
- In a second time, for a 10 harmonics signal, we vary the frequency and investigate the reconstruction error.

Stationary Case

Triangular wave synthesis

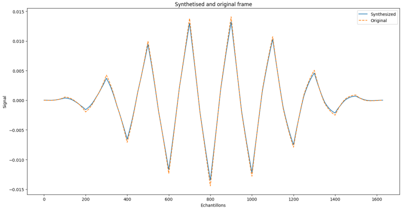


Figure: *original vs. synthesized triangular wave*

Stationary Case

Time and Relative-Time

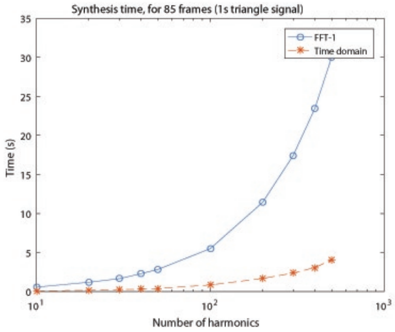


Figure: *Time of execution*

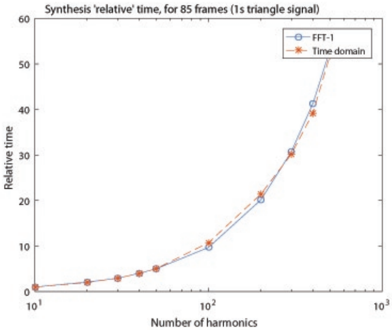


Figure: *Relative Time of Execution*

Stationary Case
RMSE

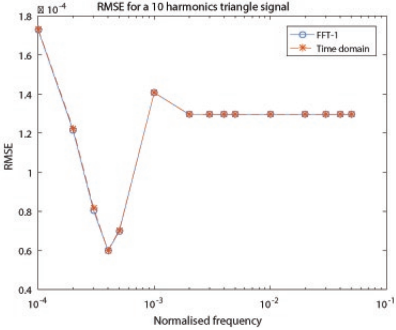


Figure: *RMSE (original vs. synthesized triangular wave*

Non-Stationary Case

Chirps

The idea is to try the method on some chirps signal. And then on real sounds, like instruments and voices.

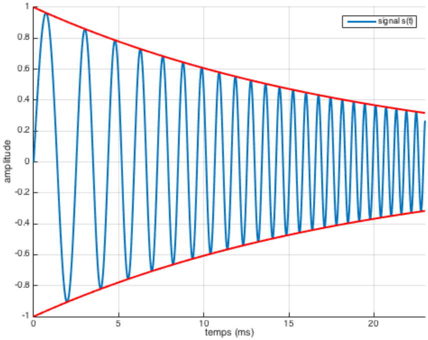


Figure: *Chirp signal to test*

We can measure the error of the lobe interpolation with the Look-Up Table. (Not done)

Part 5

Conclusion

- 6 weeks only
- Research subject $\Rightarrow$ Can it really works ?
- Lots of trouble when we try to understand the phase coherence problem

Do you have any question?

References

- Jeremy G. Wells, Damian T. Murphy “High accuracy frame-by-frame non-stationary modelling,”, Audio Lab, Intelligent Systems Group, Department of Electronics, University of York, YO10 5DD, UK, September 2006.
- Jeremy J. Wells, “Methods dor separation of amplitude and fequency modulation in Fourier transformed signals”, AudioLab, Department of Electronics, September 2010.
- D. Brandon Lloyd, Ninkunj Raghuvanshi, Naga K. Govindaraju, “Sound synthesis for impact sounds in videogames”, eXtrem computing group, Microsoft research, 2011.
- Thorsten Karrer, Eric Lee, Jan Borchers, “A phase vocoder for real-time time-stretching”, Media computing group.