

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

Content

- 1 Introduction
 - The company
 - Objective
 - Context of the Project
 - Work Environment and Project Management
- 2 Method Overview
 - Additive Synthesis (Time Domain)
 - Method Overview
- 3 The additive synthesis in frequency domain
 - Stationary Case
 - Quasi-Stationary Case
 - Non-Stationary Case
- 4 Result
 - Stationary Case
 - Non-Stationary Case
- 5 Conclusion
 - Conclusion
 - References

Part 1

- 1 Introduction
 - The company
 - Objective
 - Context of the Project
 - Work Environment and Project Management
- 2 Method Overview
 - Additive Synthesis (Time Domain)
 - Method Overview
- 3 The additive synthesis in frequency domain
 - Stationary Case
 - Quasi-Stationary Case
 - Non-Stationary Case
- 4 Result
 - Stationary Case
 - Non-Stationary Case
- 5 Conclusion
 - Conclusion
 - References

Introduction

The company



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

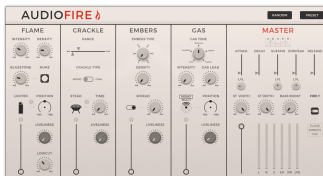


Figure: Audiofire: audio plug-in that recreates fire sound

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}

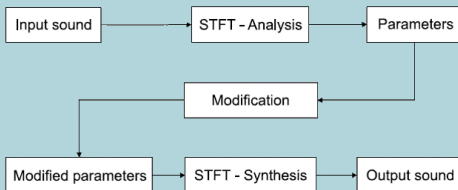


Figure: General approach for modifying a sound in the spectral domain

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

Introduction

Context of the Project

- 6 weeks only \Rightarrow 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

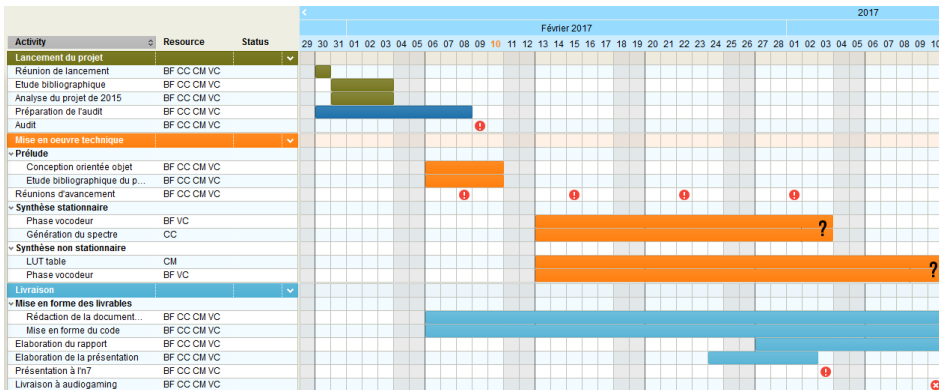
Introduction

Project Management : Gantt Chart (expected event)



Introduction

Project Management : Gantt Chart now



Part 2

- 1 Introduction
 - The company
 - Objective
 - Context of the Project
 - Work Environment and Project Management
- 2 Method Overview
 - Additive Synthesis (Time Domain)
 - Method Overview
- 3 The additive synthesis in frequency domain
 - Stationary Case
 - Quasi-Stationary Case
 - Non-Stationary Case
- 4 Result
 - Stationary Case
 - Non-Stationary Case
- 5 Conclusion
 - Conclusion
 - References

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

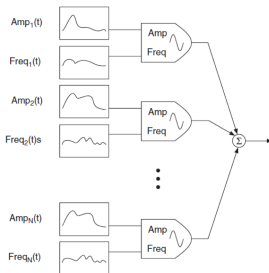


Figure: The additive synthesis

Method Overview : Windowing

Analysis

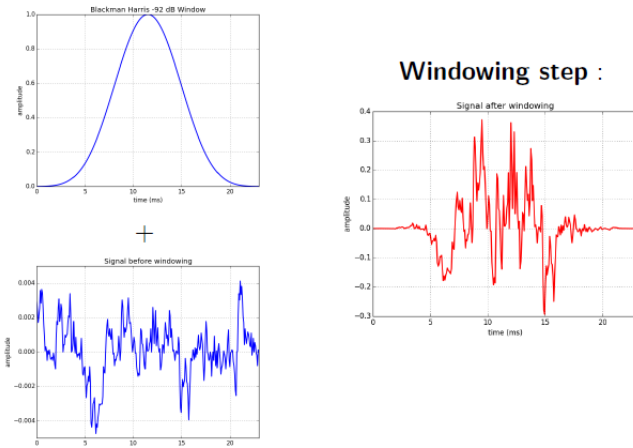


Figure: Windowing step

Method Overview : Peak detection in Frequency Domain

Analysis

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

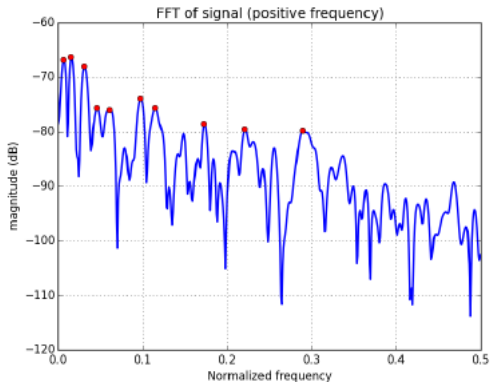


Figure: *Peak detection*

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

Synthesis

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

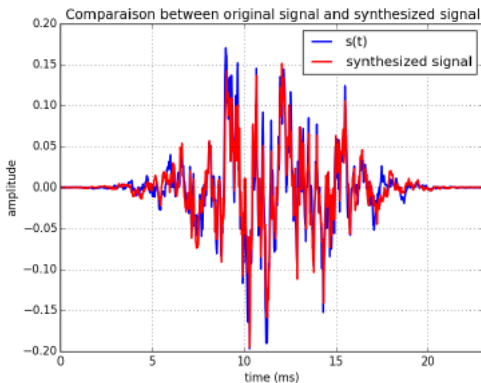


Figure: Synthesized frame vs Original frame

Part 3

- 1 Introduction
 - The company
 - Objective
 - Context of the Project
 - Work Environment and Project Management
- 2 Method Overview
 - Additive Synthesis (Time Domain)
 - Method Overview
- 3 The additive synthesis in frequency domain**
 - Stationary Case
 - Quasi-Stationary Case
 - Non-Stationary Case
- 4 Result
 - Stationary Case
 - Non-Stationary Case
- 5 Conclusion
 - Conclusion
 - References

Stationary Case

Stationary sinusoidal model

Mathematical model :

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

- 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 supposedly¹ 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 $\hat{f} \notin \left\{ \frac{2k\pi}{N} \right\}_{k=0 \dots N-1}$

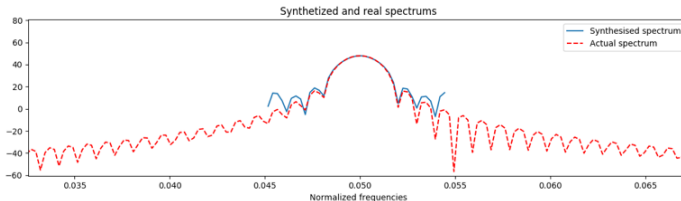


Figure: *Windowed sine lobe*

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)

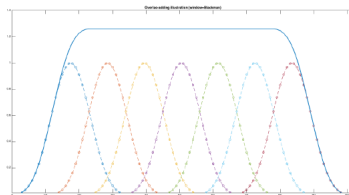


Figure: Sum of small size Hanning windows

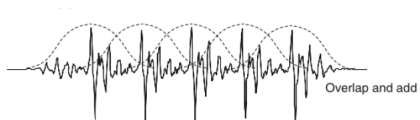


Figure: Overlap and add

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 :

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

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 !

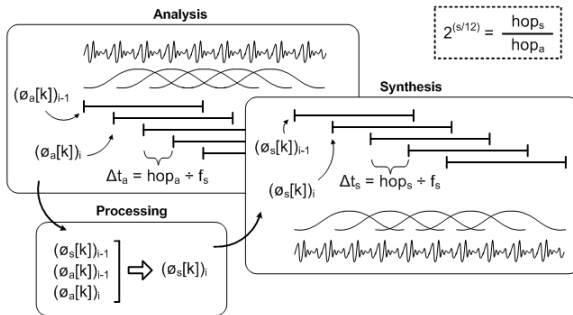


Figure: Phase Vocoder overview

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:

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

Where:

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

Hence,

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

We replace the output signal phase by this one.

Non-Stationary Case

Very different approach

Mathematical model :

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

- 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

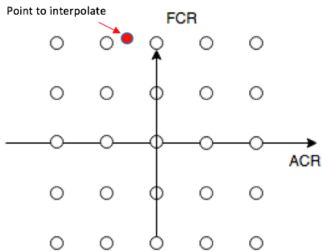


Figure: ACR/FCR grid

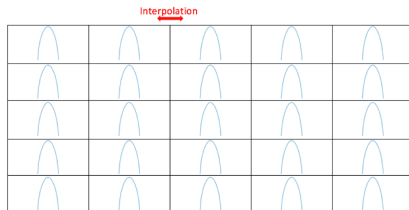


Figure: Look-up table

Non-Stationary Case

Phase Vocoder : Scaled-Phase Locking

- The idea is the same as for the stationary case, except that we have to deal with the phase locking:

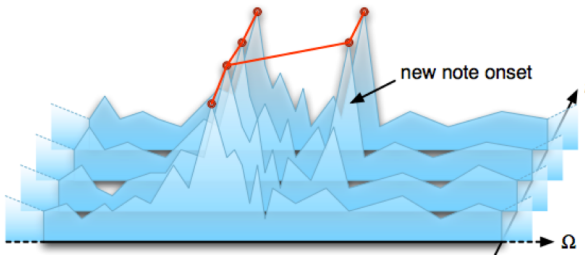


Figure: *Phase-locking scheme*

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

Part 4

- 1 Introduction
 - The company
 - Objective
 - Context of the Project
 - Work Environment and Project Management
- 2 Method Overview
 - Additive Synthesis (Time Domain)
 - Method Overview
- 3 The additive synthesis in frequency domain
 - Stationary Case
 - Quasi-Stationary Case
 - Non-Stationary Case
- 4 **Result**
 - Stationary Case
 - Non-Stationary Case
- 5 Conclusion
 - Conclusion
 - References

Stationary Case

Sine waves

Stationary Case

Triangular waves

Non-Stationary Case

Chirps

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

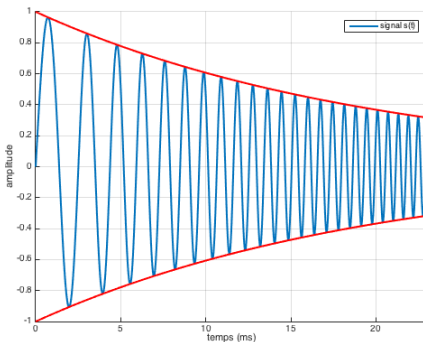


Figure: Chirp signal to test

Part 5

- 1 Introduction
 - The company
 - Objective
 - Context of the Project
 - Work Environment and Project Management
- 2 Method Overview
 - Additive Synthesis (Time Domain)
 - Method Overview
- 3 The additive synthesis in frequency domain
 - Stationary Case
 - Quasi-Stationary Case
 - Non-Stationary Case
- 4 Result
 - Stationary Case
 - Non-Stationary Case
- 5 Conclusion
 - Conclusion
 - References

Conclusion

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

References