Sound Analysis

Andrea Cuius

andrea@nocte.co.uk
www.nocte.co.uk
@q_says / @noctestudio

nocte

What is it?

Audio analysis refers to the extraction of information and meaning from audio signals for analysis, classification, storage, retrieval, synthesis, etc.

Feature extraction

A feature is a piece of information that can be extracted from the sound, it can be purely numerical like the FFT or more meaningful like the loudness or the Bark scale.

It’s usually more interesting to find informations that mean something to us. Some algorithms are based on what we perceive instead just crunched numbers.

http://en.wikipedia.org/wiki/Multimedia_Information_Retrieval#Feature_Extraction_Methods

Psychoacoustics

Psychoacoustics is the scientific study of sound perception. More specifically, it is the branch of science studying the psychological and physiological responses associated with sound.

- Loudness
- Pitch
- Bark
- Mel

Offline and real-time

Sound analysis can be organised in two main branches, **offline** and **real-time**.

Both methods can provide to similar functionalities, the offline analysis is typically more accurate(and slower), it also offers a wider range of functionalities compared with real-time techniques.

Offline sound analysis

Sound is analysed in advance and the data can be *baked* and imported in your software. If you need to develop your application based on a specific track, offline analysis is always a good option.

There are some good open source softwares available that allow you to perform almost any kind of analysis and easily export the data.

Vamp Framework

Vamp is an audio processing plugin system for plugins that extract descriptive information from audio data — typically referred to as audio analysis plugins or audio feature extraction plugins.

http://www.vamp-plugins.org/

Sonic Visualiser

Sonic Visualiser is an application for viewing and analysing the contents of music audio files.

Sonic Visualiser supports Vamp plugins:

- Queen Mary plugin set
- LibXtract plugin set

...

http://www.sonicvisualiser.org/

http://www.vamp-plugins.org/download.html

Real-time sound analysis

Sound can be processed in real-time, but with some limitations.

The most popular real-time analysis is the FFT(**Fast** Fourier Transform) which is only one of the features that can be extracted from the audio signal.

Built-in support

Cinder, OpenFrameworks, Processing and any other creative coding tool offer built in support for Audio IO and at least a few basic analysis like volume or FFT.

LibXtract

LibXtract is lightweight audio feature extraction library. It includes over 50 feature extraction.

LibXtract

https://github.com/jamiebullock/LibXtract

https://s3-eu-west-1.amazonaws.com/papers/LibXtract-_a_lightweight_feature_extraction_library.pdf

Cinder Block

https://github.com/q-depot/ciXtract

Vector and scalar features

LibXtract is organised in two sets of features:

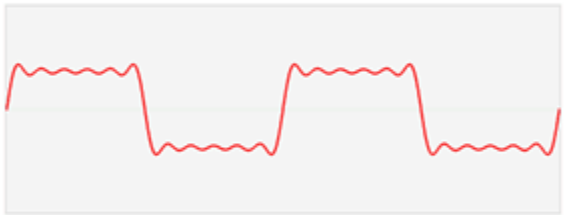
- Scalar features return one value: loudness, variation, F0...
- Vector features return a spread of data: FFT, Bark, Mel...

LibXtract: Features

MEAN, VARIANCE, STANDARD_DEVIATION, AVERAGE_DEVIATION, SKEWNESS, KURTOSIS, SPECTRAL_MEAN, SPECTRAL_VARIANCE, SPECTRAL_STANDARD_DEVIATION, SPECTRAL_SKEWNESS, SPECTRAL_KURTOSIS, SPECTRAL_CENTROID, IRREGULARITY_K, IRREGULARITY_J, TRISTIMULUS_1, TRISTIMULUS_2, TRISTIMULUS_3, SMOOTHNESS, SPREAD, ZCR, ROLLOFF, LOUDNESS, FLATNESS, FLATNESS_DB, TONALITY, CREST, NOISINESS, RMS_AMPLITUDE, SPECTRAL_INHARMONICITY, POWER, ODD_EVEN_RATIO, SHARPNESS, SPECTRAL_SLOPE, LOWEST_VALUE, HIGHEST_VALUE, SUM, NONZERO_COUNT, HPS, F0, FAILSAFE_F0, WAVELET_F0, LNORM, FLUX, ATTACK_TIME, DECAY_TIME, DIFFERENCE_VECTOR, AUTOCORRELATION, AMDF, ASDF, BARK_COEFFICIENTS, PEAK_SPECTRUM, SPECTRUM, AUTOCORRELATION_FFT, MFCC, DCT, HARMONIC_SPECTRUM, LPC, LPCC, SUBBANDS

Fast Fourier transform

A **fast Fourier transform** (**FFT**) is an algorithm to compute the discrete Fourier transform (DFT) and its inverse. A Fourier transform converts time (or space) to frequency and vice versa.



Fundamental frequency - F0

The fundamental frequency is defined as the first(frequency 0) or lowest frequency that compose the sound.

http://en.wikipedia.org/wiki/Fundamental_frequency

Bark scale

The Bark scale is a psychoacoustical scale proposed by Eberhard Zwicker in 1961. It is named after Heinrich Barkhausen who proposed the first subjective measurements of loudness. The scale ranges from 1 to 24 and corresponds to the first 24 critical bands of hearing. It is related to, but somewhat less popular than the mel scale.

http://en.wikipedia.org/wiki/Bark_scale

Mel scale

The mel scale, named by Stevens, Volkman and Newman in 1937 is a perceptual scale of pitches judged by listeners to be equal in distance from one another.

http://en.wikipedia.org/wiki/Mel_scale

http://en.wikipedia.org/wiki/Mel-frequency_cepstrum

ciXtract - LibXtract Cinder block

```
#include "ciXtract.h"

class BasicApp : public AppNative {
	…
	ciXtractRef		mXtract;
}

void BasicApp::setup() {
	mXtract			= ciXtract::create();			// create an instance
	mFeatures		= mXtract->getFeatures();		// get all available features
	mXtract->enableFeature(XTRACT_SPECTRUM);		// enable the features
}
```

ciXtract - LibXtract Cinder block

LibXtract needs to be updated on each frame, the update() method ensure that all the calls are done in the right order, some features depends on each other and before evaluating one, it evaluates all its dependencies.

```
void BasicApp::update() {
	mXtract->update();			// update LibXtract(process data)
	...
}
```

ciXtract - LibXtract Cinder block

```
void BasicApp::drawData( ciXtractFeatureRef feature, … ) {
    std::shared_ptr<double> data  = feature->getResult();      // the data
    int         dataN    = feature->getResultN();              // data size
    float       min      = feature->getResultMin();            // min value
    float       max      = feature->getResultMax();            // max value
    ...
    for( int i = 0; i < dataN; i++ )  {
        // do something (awesome) …
    }
}
```

ciXtractReceiver

ciXtractReceiver is a Cinder block to quickly implement an OSC receiver that connects with XtractSenderOSC which is the bit that perform the audio analysis.

XtractSenderOSC comes with an XML configuration file to enable/disable any available feature, configure OSC and specify the audio source.

https://github.com/q-depot/ciXtractReceiver

ciXtractReceiver

```
#include "ciXtractReceiver.h"

class BasicApp : public AppNative {
	…
	ciXtractReceiverRef	mXtractReceiver;
}

void BasicApp::setup() {
	mXtract = ciXtractReceiver::create();		// create an instance
}
```

ciXtractReceiver

```
void BasicApp::update() {
	mXtract->update();
}

void BasicApp::draw() {
	FeatureDataRef feature;
	feature = mXtract->getFeatureData( mAvailableFeatures[k] );
	ciXtractReceiver::drawData( feature, rect );		// utility to draw the data
}
```

Andrea Cuius

andrea@nocte.co.uk
www.nocte.co.uk
@q_says / @noctestudio

nocte