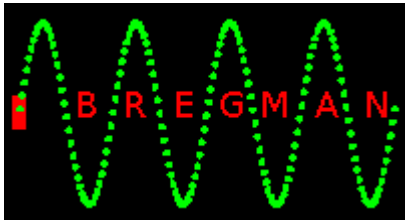

Bregman Audio-Visual Information Toolbox

Advanced Tools for the Digital Arts and Humanities

Introduction

Welcome to the Bregman toolbox from the Bregman Music and Audio Research Studio at Dartmouth College. These pages document the toolbox and provide a set of guided tutorials to its use.

Who is Bregman for?

Bregman provides Python tools to support research and development including, but not limited to:

- Music IR - explore methods for audio and music machine learning and information retrieval
- Multimedia IR - explore methods for video information retrieval (requires OpenCV package)
- Music Cognition/Neuroscience - classify/predict music evoked fMRI and EEG signals (requires PyMVPA package)
- Computational Musicology - run queries on music collections
- Teaching (e.g. Music Information Retrieval) - hands-on tools for undergraduate, Masters, and Ph.D. students
- Application development - make new Python-based MIR applications such as music search and recommender systems

Libraries and Dependencies

- Required: pylab (numpy + scipy + matplotlib).
- Linux and OSX users with package management tools (e.g. easy_install, dpkg, apt-get, ports, yum), should install the unix packages from their distributions, e.g. (using apt-get):

```
sudo apt-get install ipython python-numpy python-matplotlib python-scipy
```

- Windows, and OSX without package manager, install the Enthought Python Distribution (EPD). Available for FREE for academic use.
- Strongly recommended: instal scikits.audiolab for audio play and wavread/wavwrite functions.

Download Bregman

Save the ZIP file Bregman Toolbox Version 0.12-09.15 to your machine.

Installation

unzip the installer directory. You will need to install as administrator. In a UNIX terminal:

cd /path/to/installer/directory sudo python setup.py install

If you do not have admin (sudo) privileges, you can install in your home path using the –prefix “$HOME” option to setup.py

Tutorial 0: Getting Started

First launch the *ipython* shell with *pylab* preloaded:

```
ipython --pylab # launch the ipython shell with pylab
```

Import the entire bregman toolbox and specify an audio file to work with:

```
from bregman.suite import *

#use built-in audio examples in audio_dir
audio_file = os.path.join(audio_dir,"gmin.wav")
```

Extract short-time Fourier transform, specifying window parameters:

```
linspec = LinearFrequencySpectrum(audio_file, nfft=1024, wfft=512, nhop=256)
linspec.feature_plot(dbscale=True)
title('Wide-band Linear Spectrum')
```

Play the audio_file using the built-in play() command”:

```
x,sr,fmt = wavread(audio_file) # load the audio file
play(x, sr) # play it
```

Invert the short-time Fourier transform back to audio using the feature inverse() method.

```
x_hat = linspec.inverse(usewin=0) # invert features to audio (use original phases, no windowing)
play(x_hat)
```

Extract the log-frequency spectrum, specifying windowing parameters:

```
logspec = LogFrequencySpectrum(audio_file, nhop=2205) # extract log spectrum
logspec.feature_plot(dbscale=True) # plot features on dB scale
title('Narrow-band Log Spectrum')
```

Invert the log spectrum using the feature inverse() method. The log-frequency spectrum does not contain complete information so we'll need to estimate the phases, via the *pvoc=True* flag, and use a reconstruction window, *usewin=True*. The signal should also be balanced to ensure no clipping on audio output.

```
x_hat = logspec.inverse(pvoc=True) # invert phaseless features to audio
play(balance_signal(x_hat),sr) # play inverted features
```

Inspect the default feature parameters and features module help:

```
# list the (default) parameters that control feature extraction."
Features.default_params() # inspect default parameters

help(features) # see help on the features module
```

List the tutorials and run one:

```
# show list of tutorials:
get_tutorials()

# execute the first tutorial (1. features)
execfile(get_tutorials()[1])
```

Tutorials

Bregman comes with a set of tutorials that we recommend you become familiar with.

- Tutorial1 - Audio feature extraction and visualization.
- Tutorial2 - Audio test signal synthesis.
- Tutorial3 - Audio similarity analysis
- Tutorial4 - Concatenative audio synthesis with a source and target.
- Tutorial5 - Audio source separation using non-negative matrix factorization.

Bregman Python Modules

- suite - wrapper package to bundle all the bregman tools
- testsignal - test signal generators
- features - feature extractors and visualizers
  - Overview
  - Features instance members
  - Feature Extractors
- segment - media segmentation and segmented feature extraction
- psychoacoustics - perceptual methids, critical bands, loudness scales
- tuning - methods for generating different tunings, temperaments, and scales
- distance - distance metrics, dynamic time-warping, and multidimensional scaling
- classifier - unsupervised and supervised learning with feature data
- pyadb - interface to audiodb feature-vector database and scalable content-based retrieval
- audiodb - audiodb extensions helper class and feature regularization / pre-processing
- audiocollection - manage collections of audio and features with audiodb
- metadata - tools for searching and manipulating metadata from the Web
- testcollection - generate signals and evaluate features and search algorithms
- evaluate - general-purpose evaluation module using user-supplied ground-truth
- sound - audio input/output utilities and sound-file read/write methods
- lsh - fast (sublinear time complexity) search using locality sensitive hashing

Indices and tables

- *Index*
- *Module Index*
- *Search Page*

Funding sources:

NATIONAL ENDOWMENT FOR THE Humanities