higrid

Release 0.1.0

Apr 11, 2019

Contents

1 HiGRID documentation **1**
- 1.1 Installation . . . 1
- 1.2 Example usage . . . 2

2 Indices and tables **11**

Python Module Index **13**

CHAPTER 1

HiGRID documentation

higrid is a Python package that implements the hierarchical grid refinement (HiGRID) direction-of-arrival estimation algorithm for rigid spherical microphone arrays. The algorithm is based on the calculation of steered response power density (SRPD) maps, and spatial entropy-based (multiple) peak detection.

The method was developed by the researchers in METU Spatial Audio Research Group http://www.sparglab.org. The technical details are available in the following papers. Please cite our papers if you want to use the code and/or the data provided in this package in your research.

Coteli, M. B., Olgun, O., and Hacihabiboglu, H. (2018). Multiple Sound Source Localization With Steered Response Power Density and Hierarchical Grid Refinement. IEEE/ACM Trans. Audio, Speech and Language Process., 26(11), 2215-2229. https://ieeexplore.ieee.org/document/8418732

Olgun, O. and Hacihabiboglu, H., (2018) "Localization of Multiple Sources in the Spherical Harmonic Domain with Hierarchical Grid Refinement and EB-MUSIC". In 2018 16th Int. Workshop on Acoust. Signal Enhancement (IWAENC-18) (pp. 101-105), Tokyo, Japan. https://ieeexplore.ieee.org/abstract/document/8521365 ; also available at https://bit.ly/2I81eru

1.1 Installation

Due to one of the key external packages (i.e. healpy) working on Linux and MacOS only (and not Windows), higrid will also only work on Linux and MacOS, only. The current version was tested on Ubuntu 18.04.2 LTS and MacOS 10.14.1 with Python 3.6.1.

1.1.1 Prerequisites

In order to install and use the *higrid* you need Python 3.x on your system. A few tweaks would be necessary to make it work with Python 2.7 (incompatibilities are due to scipy version, as well as default parameters for pickle implementation in 2.7 and defaultdict member functions in 2.7). The following packages are also needed:

- numpy http://www.numpy.org
- scipy https://www.scipy.org

- healpy https://github.com/healpy/healpy
- PeakUtils https://bitbucket.org/lucashnegri/peakutils
- tqdm https://www.github.com/tqdm/tqdm
- madmom https://github.com/CPJKU/madmom

Please also check the requirements.txt file at https://github.com/metu-sparg/higrid/blob/master/requirements.txt for the specific version requirements.

1.1.2 Installation from *pip3* Package

Using *pip3* is the easiest way to install *higrid* :

```
pip3 -install higrid
```

1.2 Example usage

Please see the file 'higrid_example.py' for a more complete example.

```
import higrid as hg

testinstance = set([(1, 1, 2), (5, 1, 2), (5, 5, 2), (1, 5, 2)])
sg = hg.composescene(
     ['music/mahler_vl1a_6.wav', 'music/mahler_vl1b_6.wav', 'music/mahler_vl2a_6.wav',
 'music/mahler_vl2b_6.wav'],
     testinstance, (0, 192000))
pd = hg.loadpixbasis()
th, ph = hg.higridestimate(sg, pd, maxnum=1000, Fs=48000, Ndec=4, NFFT=1024, olap=16,
tLevel=3, dpdflag=True,
                           fL=2608., fH=5216., thr=6)
```

1.2.1 higrid

higrid package

higrid consists of the following modules:

Microphone module

Microphone class that is to be expanded in the future versions of the package.

class Microphone.**EigenmikeEM32**

Bases: *Microphone.MicrophoneArray*

Eigenmike em32 class that inherits from the MicrophoneArray class.

returnAsStruct()

Returns the attributes of the Eigenmike em32 as a struct

Returns dict object with the name, type, thetas, phis, radius, weights, version, numelements, directivity, info fields

class Microphone.**Microphone**(*name='Generic'*, *version='1.0'*, *direct='Omnidirectional'*)
Bases: object

Microphone class

getname()
Getter for the name

Returns Name (str) of the microphone object

getversion()
Getter for the version

Returns Version (str) of the microphone object

setname(*name*)
Setter for the name attribute

Parameters **name** – Name of the microphone (str)

setversion(*version*)
Setter for the version attribute

Parameters **name** – Version of the microphone (str)

class Microphone.**MicrophoneArray**(*name*, *typ*, *version*, *direct*)
Bases: *Microphone.Microphone*

MicrophoneArray Class that inherits from the Microphone class

gettype()
Getter for array type

Returns Type of the array (str)

settype(*typ*)
Setter for array type

Parameters **typ** – Type of the array (str)

TreeClusteri module

Submodule containing functions used in clustering multiresolution SRPD maps

TreeClusteri.**hpneighbouridx**(*level*, *idx*)
Neighbours of a Healpix pixel (in nested format)

Parameters

- **level** – Resolution level
- **idx** – Index of the Healpix pixel

Returns Neighbouring nodes of the pixel

TreeClusteri.**neighboursofset**(*cset*)
Return all the neighbours of a set of Healpix pixels (in nested format) :param cset: :return:

TreeClusteri.**preprocess**(*trr*, *tLevel*)
Eliminate leaf nodes that are below the mean

Parameters

- **trr** – Healpix tree (as a defaultdict)

- **tLevel** – Level at which the leaf nodes are

Returns Tree as a defaultdict after mean thresholding

TreeClusteri.**treeCluster**(*trr*, *tLevel*)
Neighbouring Nodes Labelling (NNL) for clustering pixels

Parameters

- **trr** – Healpix tree (as a defaultdict)
- **tLevel** – Level at which the leaf nodes are

Returns Healpix pixel clusters that belong to a single source

Treei module

Tree selection based on spatial entropy

dpd module

Submodule including functions used for the direct path dominance (DPD) test

emulate module

Submodule used for creating emulated recordings using acoustic impulse responses

emulate.**combinescene**(*sg1*, *sg2*)
Linearly combines two scenes pertaining to em32 recordings

Parameters

- **sg1** – Scene 1 (32 x samples numpy array)
- **sg2** – Scene 2 (32 x samples numpy array)

Returns Combined scene (32 x samples numpy array)

emulate.**composescene**(*filelist*, *dirset*, *samples=(0*, *96000)*, *roomstr='ii-s05'*)
Compose an emulated scene using a number of anechoic sound signals and measured AIRs

Parameters

- **filelist** – List of files to be used
- **dirset** – Set containing tuples with (X, Y, Z) as the AIR indices
- **samples** – Start and end points of samples to be prococessed as a tuple (sstart, send)
- **roomstr** – Used to select from a specific directory (default is 'ii-s05' as we only provided AIRs for that room)

Returns 32 channels of audio from an emulated em32 recording.

emulate.**emptyscene**(*sha=(32*, *48000)*)
Returns an empty scene

Parameters **sha** – Tuple containing number of channels and number of samples (nchan, samples) (default = (32, 48000))

Returns An empty scene containing nchan channels and the given number of samples (empty numpy array)

emulate.**emulatescene**(*insig*, *gain*, *irspath*)
Emulates a scene by convolving a source input signal with em32 AIRs

Parameters

- **insig** – (Single-channel) input signal
- **gain** – Gain (scalar)
- **irspath** – Path to the AIRs to be used

Returns 32 channels of audio from an emulated em32 recording.

emulate.**realrec**(*dirpath*, *prefix*, *samples*)
Create a scene from real em32 recordings

Parameters

- **dirpath** – Path containing the em32 recordings
- **prefix** –
- **samples** – Number of samples to use

Returns 32 channel em32 recording

higridestimate module

Main submodule that includes functions used in DOA estimation with the HiGRID algorithm

shd module

Submodule that includes functions related to spherical harmonic decomposition (SHD)

testpos3d module

Submodule for selecting emulated source positions

testpos3d.**angle**(*tupl1*, *tupl2*, *mtupl*)
Returns the angle between two vectors

Parameters

- **tupl1** – Tuple containing the coordinates of a point
- **tupl2** – Tuple containing the coordinates of another point
- **mtupl** – Tuple containing the coordinates of origin

Returns Angle between the two vectors (in radians)

testpos3d.**histclust**(*drrmat*, *dind*, *N*, *numthr*)
Returns a number of clusters of sources positions based on their D/R ratios

Parameters

- **drrmat** – Matrix containing coordinates of source oposition on the grid and its D/R ratio
- **dind** – Histogram bin to choose

- **N** – Number of bins to use
- **numthr** – At least this many sources in a given bin has to be present

Returns

testpos3d.**measgrid**()
Returns the measurement positions and DRR values for emulations

Returns Defaultdict containing tuple (xindex, yindex, zindex) as keys and DRR as values

testpos3d.**select**(*grd*, *n=4*, *dind=3*, *dclust=8*, *thcond=<MagicMock name='mock.__truediv__()' id='140539644536144'>*)
Returns a single test scene instance

Parameters

- **grd** – Measurement grid from measgrid()
- **n** – Number of sources to choose (default = 4)
- **dind** – Cluster index (default = 3)
- **dclust** – Number of bins to use in histogram for D/R ratio clustering (default = 8)
- **thcond** – Minimum angle between consecutive sources (in radians) (default = pi/4)

Returns One test instance and the average D/R ratio at the selected positions

testpos3d.**selectset**(*grd*, *n=4*, *dind=3*, *dclust=8*, *thcond=<MagicMock name='mock.__truediv__()' id='140539641887824'>*, *trialcount=1*)
Returns a number of test scene instances with similar D/R ratio and a prescribed minimum angle between them

Parameters

- **grd** – Measurement grid from measgrid()
- **n** – Number of sources to choose (default = 4)
- **dind** – Cluster index (default = 3)
- **dclust** – Number of bins to use in histogram for D/R ratio clustering (default = 8)
- **thcond** – Minimum angle between consecutive sources (in radians) (default = pi/4)
- **trialcount** – Number of randomly selected scenes (default = 1)

Returns List containing trialcount number of test instances and the average D/R ratio at the selected positions

testpos3d.**vdist**(*a*, *b*)
Returns the angle between two vectors

Parameters

- **a** – Vector 1
- **b** – Vector 2

Returns Normalised angle [-1,+1] between the two vectors

treeutils module

Submodule including some utility functions for manipulating tree representation

treeutils.**children**(*level*, *idx*)
Return all the children of the Healpix pixel idx at level (in nested format)

Parameters

- **level** – Resolution level
- **idx** – Pixel index

Returns All the parents of the pixel

treeutils.**parent**(*level*, *idx*)
Return the parent of a given healpix pixel (in nested format)

Parameters

- **level** – Resolution level
- **idx** – Index at the given level

Returns Tuple (lvl, idp) including the level and the index of the parent pixel

treeutils.**parents**(*level*, *idx*)
Return all the (grand-)parents of the Healpix pixel idx at level (in nested format)

Parameters

- **level** – Resolution level
- **idx** – Pixel index

Returns All the parents of the pixel

treeutils.**siblings**(*level*, *idx*)
Return the siblings of the Healpix pixel idx at level (in nested format)

Parameters

- **level** – Resolution level
- **idx** – Pixel index

Returns Return the siblings of the pixel

utils module

Submodule including some general utility functions

utils.**cart2sph**(*x*, *y*, *z*)
r, th, ph = cart2sph(x, y, z)

Return the spherical coordinate representation of point(s) given in Cartesian coordinates

As usual r is the radius, th is the elevation angle defined from the positive z axis and ph is the azimuth angle defined from the positive x axis

utils.**fulltree**(*tLevel=3*, *val=0*)
Create a defaultdict containing all the pixels in a healpix grid, initialised with the default value

Parameters

- **tLevel** – Healpix resolution level (default = 3)
- **val** – Value of each pixel (default = 0)

Returns defaultdict containing all the pixels in the healpix grid at the given resolution

utils.**histtotree**(*H*, *the*, *phe*, *tLevel*)
Convert 2D DOA histogram to a healpix tree representation

Parameters

- **H** – 2D DOA histogram matrix
- **the** – Array of azimuth angles corresponding to columns of the 2D histogram matrix
- **phe** – Array of inclination angles corresponding to columns of the 2D histogram matrix
- **tLevel** – Healpix resolution level (default = 3)

Returns defaultdict containing healpix grid st the given resolution containing the 2D histogram

utils.**loadpixbasis**()
Returns the

Returns

utils.**node2vec**(*level*, *idx*)
Converts the centre direction of a Healpix pixel to a unit vector

Parameters

- **level** – Resolution level
- **idx** – Index of the pixel

Returns 3x1 array containing the unit vector

utils.**processirs**(*irs*, *monosnd*)
Returns an emulated recording of em32 using acoustic impulse responses and an anechoic sound signal

Parameters

- **irs** – Acoustic impulse responses obtained using em32
- **monosnd** – Monophonic sound signal to be convolved with the AIRs

Returns 32-channels emulated recording of em32 as a numpy array

utils.**selectbinindx**(*fidx*, *tidx*, *Pnm*, *Ndec=4*)
Returns the SHD vectors of a given time-frequency bin

Parameters

- **fidx** – Frequency index
- **tidx** – Time index
- **Pnm** – List of SHD-STFT matrices
- **Ndec** – SHD order (default = 4)

Returns SHD vector, SHD order

utils.**selectsome**(*idx*, *idy*, *maxnum*)
Randomly selects a given number of time-frequency bins

Parameters

- **idx** – List containing frequency indices of selected bins
- **idy** – List containing time indices of selected bins
- **maxnum** – Maximum number of bins to select (in None, return all indices)

Returns

utils.**sph2cart**(*r*, *th*, *ph*)
Converts vector in spherical coordinates to Cartesian coordinates

Parameters

- **r** – Radius
- **th** – Azimuth angle
- **ph** – Inclination angle

Returns Vector in Cartesian coordinates

utils.**sph_jnyn**(*N*, *kr*)

Returns spherical Bessel functions of the first (jn) and second kind (yn) and their derivatives

Parameters

- **N** – Function order
- **kr** – Argument

Returns jn, jn', yn, yn'

NOTE: Emulates the behaviour of sph_jnyn() in early versions of scipy (< 1.0.0).

utils.**wavread**(*wave_file*)

Returns the contents of a wave file

Parameters **wave_file** – Path to the wave_file to be read

Returns (signal, sampling rate, number of channels)

NOTE: Wavread solution was adapted from https://bit.ly/2Ubs9Jp

Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

emulate, 4

m

Microphone, 2

t

testpos3d, 5
TreeClusteri, 3
treeutils, 6

u

utils, 7

Index

A

angle() (*in module testpos3d*), 5

C

cart2sph() (*in module utils*), 7
children() (*in module treeutils*), 6
combinescene() (*in module emulate*), 4
composescene() (*in module emulate*), 4

E

EigenmikeEM32 (*class in Microphone*), 2
emptyscene() (*in module emulate*), 4
emulate (*module*), 4
emulatescene() (*in module emulate*), 5

F

fulltree() (*in module utils*), 7

G

getname() (*Microphone.Microphone method*), 3
gettype() (*Microphone.MicrophoneArray method*), 3
getversion() (*Microphone.Microphone method*), 3

H

histclust() (*in module testpos3d*), 5
histtotree() (*in module utils*), 7
hpneighbouridx() (*in module TreeClusteri*), 3

L

loadpixbasis() (*in module utils*), 8

M

measgrid() (*in module testpos3d*), 6
Microphone (*class in Microphone*), 2
Microphone (*module*), 2
MicrophoneArray (*class in Microphone*), 3

N

neighboursofset() (*in module TreeClusteri*), 3
node2vec() (*in module utils*), 8

P

parent() (*in module treeutils*), 7
parents() (*in module treeutils*), 7
preprocess() (*in module TreeClusteri*), 3
processirs() (*in module utils*), 8

R

realrec() (*in module emulate*), 5
returnAsStruct() (*Microphone.EigenmikeEM32 method*), 2

S

select() (*in module testpos3d*), 6
selectbinindx() (*in module utils*), 8
selectset() (*in module testpos3d*), 6
selectsome() (*in module utils*), 8
setname() (*Microphone.Microphone method*), 3
settype() (*Microphone.MicrophoneArray method*), 3
setversion() (*Microphone.Microphone method*), 3
siblings() (*in module treeutils*), 7
sph2cart() (*in module utils*), 8
sph_jnyn() (*in module utils*), 9

T

testpos3d (*module*), 5
treeCluster() (*in module TreeClusteri*), 4
TreeClusteri (*module*), 3
treeutils (*module*), 6

U

utils (*module*), 7

V

vdist() (*in module testpos3d*), 6

W

wavread() (*in module utils*), 9