audio_synch_tool Documentation

Release 0.5.0

Andres FR

May 24, 2019

CONTENTS:

CHAPTER

ONE

AUDIO_SYNCH_TOOL PACKAGE

1.1 Subpackages

1.1.1 audio_synch_tool.data package

1.1.1.1 Module contents

1.2 Submodules

1.3 audio_synch_tool.mvn module

This module contains functionality concerning the adaption of the XSENS MVN format into our Python setup.

The following section introduces the contents of the imported MVN file and the way they can be accessed from Python:

```
# load mvn schema https://www.xsens.com/mvn/mvnx/schema.xsd
MVN_SCHEMA_PATH = "xxx"
mvn_path = "yyy"
mmvn = Mvn(mvn_path, MVN_SCHEMA_PATH)

# These elements contain some small metadata:
mmvn.mvn.attrib
mmvn.mvn.comment.attrib
mmvn.mvn.securityCode.attrib["code"]
mmvn.mvn.subject.attrib

# subject.segments contain 3D pos_b labels:
for ch in mmvn.mvn.subject.segments.iterchildren():
    ch.attrib, [p.attrib for p in ch.points.iterchildren()]

# Segments can look as follows: ``['Pelvis', 'L5', 'L3', 'T12', 'T8', 'Neck',
'Head', 'RightShoulder', 'RightUpperArm', 'RightForeArm', 'RightHand',
'LeftShoulder', 'LeftUpperArm', 'LeftForeArm', 'LeftHand', 'RightUpperLeg',
'RightLowerLeg', 'RightFoot', 'RightToe', 'LeftUpperLeg', 'LeftLowerLeg',
'LeftFoot', 'LeftToe']``

# sensors is basically a list of names
for s in mmvn.mvn.subject.sensors.iterchildren():
    s.attrib

#  Joints is a list that connects segment points:
```

(continues on next page)

(continued from previous page)

```
for j in mmvn.mvn.subject.joints.iterchildren():
    j.attrib["label"], j.getchildren()

# miscellaneous:
for j in mmvn.mvn.subject.ergonomicJointAngles.iterchildren():
    j.attrib, j.getchildren()

for f in mmvn.mvn.subject.footContactDefinition.iterchildren():
    f.attrib, f.getchildren()

# The bulk of the data is in the frames.
print("frames_metadata:", mmvn.frames_metadata)
print("first_frame_type:", mmvn.config_frames[0]["type"])
print("normal_frames_type:", mmvn.normal_frames[0]["type"])
print("num_normal_frames:", len(mmvn.normal_frames))

# Metadata looks like this:
{'segmentCount': '23', 'sensorCount': '17', 'jointCount': '22'}

# config frames have the following fields:
['orientation', 'position', 'time', 'tc', 'ms', 'type']

# normal frames have the following fields:
['orientation', 'position', 'velocity', 'acceleration',
 'angularVelocity', 'angularAcceleration', 'footContacts',
 'sensorFreeAcceleration', 'sensorMagneticField', 'sensorOrientation',
 'jointAngle', 'jointAngleXZY', 'jointAngleErgo', 'centerOfMass', 'time',
 'index', 'tc', 'ms', 'type']
```

The following fields contain metadata about the frame:

time ms since start (integer). It is close to `int(1000.0 * index / samplerate)`, being equal most of the times and at most 1 milisecond away. It is neither truncated nor rounded, maybe it is given by the hardware.

index starts with 0, +1 each normal frame

tc string like '02:23:28:164'

ms unix timestamp like 1515983008686 (used to compute time)

type one of "identity", "tpose", "tpose-isb", "normal"

# The following fields are float vectors of the following dimensionality:

orientation `segmentCount*4 = 92`

position, velocity, acceleration, angularVelocity, angularAcceleration `segmentCount*3 = 69`

footContacts `4`

sensorFreeAcceleration, sensorMagneticField `sensorCount*3 = 51`

sensorOrientation `sensorCount*4 = 68`

jointAngle, jointAngleXZY `jointCount*3 = 66`

jointAngleErgo `12`

centerOfMass `3`

class `audio_synch_tool.mvn.`**`Mvn`**(*mvn_path*, *schema_path=None*)

Bases: `object`

This class imports and adapts an XML file (expected to be in MVN format) to a Python-friendly representation. See this module's docstring for usage examples and more information.

`get_normalframe_magnitudes`()

Returns A list of the magnitude names in each of the `self.normal_frames`

`get_normalframe_sequences`(*device='cpu'*)

Parameters

- **`magnitude`** (*`str`*) – One of `self.get_segments()`
- **`magnitude`** – One of `self.get_normalframe_magnitudes()`

Returns a torch tensor of shape `(num_normalframes, num_channels)` where the number of channels is e.g. 1 for scalar magnitudes, 3 for 3D vectors...

`get_segments`()

Returns A list of the segment names in `self.mvn.subject.segments`, ordered by id (starting at 1 and incrementing +1).

1.4 audio_synch_tool.plotters module

class `audio_synch_tool.plotters.`**`MultipleDownsampledPlotter1D`**(*arrays*, *samplerates=None*, *max_datapoints=10000*, *shared_plots=None*)

Bases: `object`

This class generates a matplotlib figure that plots several 1-dimensional arrays. Since some arrays can be quite long, it features a built-in downsampling mechanism that plots a (configurable) number of points at most. The downsampling mechanism is based on this code:

https://matplotlib.org/3.1.0/gallery/event_handling/resample.html

Note: The downsampling mechanism only affects the display, not the data, and is refreshed every time the user changes the perspective. If the user zooms close enough the data will be displayed in its original form. Therefore downsampling helps to provide a quicker interaction when scrolling large files, while allowing for sample-precise inspection.

`FIG_ASPECT_RATIO = (10, 8)`

`FIG_MARGINS = {'bottom': 0.1, 'hspace': 0.5, 'left': 0.1, 'right': 0.95, 'top': 0`

`make_fig`(*num_xticks=15*, *num_yticks=10*, *tick_fontsize=7*, *tick_rot_deg=15*, *num_decimals=3*, *show_idx=True*)

Returns A matplotlib Figure containing the array given at construction. The interactive plot of the figure will react to the user's zooming by showing approximately `self.max_datapoints` number of samples.

1.5 audio_synch_tool.utils module

class `audio_synch_tool.utils.`**`DownsamplableFunction`**(*arr*, *max_datapoints*)

Bases: `object`

Encapsulates the downsampling functionality to prevent side effects, and reduce code verbosity in plotter.

`downsampled`(*xstart*, *xend*)

This function performs downsampling by reading one sample every `(xend-xstart)//max_datapoints` from x_vals and y_vals.

class `audio_synch_tool.utils.`**`SampleToTimestampFormatter`**(*samplerate*, *num_decimals=3*, *show_idx=True*)

Bases: `object`

This functor can be passed to `plt.FuncFormatter` to generate custom tick labels. It fulfills the interface (val, pos) -> str.

Specifically, converts `val` number representing a sample into a string showing the corresponding elapsed time since sample 0, asuming the given samplerate. Usage example:

```
ax.xaxis.set_major_formatter(plt.FuncFormatter(
                             SampleToTimestampFormatter(sr)))
```

class `audio_synch_tool.utils.`**`SharedXlimCallbackFunctor`**(*functors*)

Bases: `object`

If multiple axes share

class `audio_synch_tool.utils.`**`Timestamp`**(*sample_nr*, *samplerate*)

Bases: `object`

`as_tuple`()

Returns the tuple of integers (days, hours, mins, secs, microsecs)

`days`

`hours`

`microsecs`

`mins`

`sample_nr`

`samplerate`

`secs`

`total_seconds`

class `audio_synch_tool.utils.`**`XlimCallbackFunctor`**(*axis*, *lines*, *arrays*)

Bases: `object`

Encapsulates the downsampling callback functionality to prevent side effects. Instances of this functor can be passed to an axis like this:

```
``ax.callbacks.connect('xlim_changed',
                  DownsamplingCallbackFunctor(ax, [line1..], [arr1...]))``
```

1.6 Module contents

CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX