opti_ssr Documentation

Release 0.1.4

Felix Immohr

Jun 13, 2018

Contents

This python package provides the necessary tools to use an OptiTrack optical tracking system for different applications of the SoundScape Renderer (SSR) including listener position and orientation tracking in local sound field synthesis.

It contains several modules including a module to connect to the OptiTrack system (opti_client) and a seperate one to connect to and control instances of the SSR (ssr_client). The module that connects these aforementioned ones and implements sequence and desired functionality is also part of the package (bridges).

Note that the optirx 1.10 library is included here with only minor changes from the original source and that the modules ssr_client and opti_client respectively are designed to be used independently in other projects as well.

Documentation: http://opti-ssr.rtfd.io/

Source code: https://github.com/OptiTools/opti_ssr

Python Package Index: http://pypi.python.org/pypi/opti_ssr/

Example files to set up the SSR: https://github.com/OptiTools/opti_ssr-examples

License: MIT – see the file `LICENSE` for details.

CHAPTER 1

Installation

Aside from Python itself, NumPy and pyquaternion are needed. It should work with both Python3 as well as Python2.

The easiest way to install this package is using pip to download the latest release from PyPi:

```
pip install opti_ssr
```

CHAPTER 2

Usage

To use opti_ssr you can use a demo function like the basic one below. Simply instantiate the necessary class objects according to the given parameters and start the thread in the class that contains the functionality. Ready-to-use demo functions to demonstrate head orientation tracking in binaural synthesis or listener tracking in local sound field synthesis are available at the github repository. The SoundScape Renderer has to be started prior to opti_ssr.

```
import opti_ssr

def demo(ssr_ip, ssr_port, opti_unicast_ip, opti_multicast_ip, opti_port, ssr_end_
↪message):
    optitrack = opti_ssr.OptiTrackClient(opti_unicast_ip, opti_multicast_ip, opti_
↪port)
    ssr = opti_ssr.SSRClient(ssr_ip, ssr_port, ssr_end_message)
    headtracker = opti_ssr.HeadTracker(optitrack, ssr)
    headtracker.start()

if __name__ == "__main__":
    demo()
```

CHAPTER 3

Contents:

3.1 LocalWFS (Listener Tracking)

A python module for demonstrating listener tracking in local wave field synthesis.

To accomplish the listener tracking two instances of the SoundScape Renderer (SSR) are necessary. The first SSR instance shifts a the reference position of a real reproduction setup in relation to a virtual point source array placed around the listener at the center. The second SSR instance shifts the reference position of aforementioned point sources as the virtual reproduction setup in relation to the real sources based on audio files.

Usage: python opti_ssr_demo.py [SSR_IP] [SSR1 port] [SSR2 port] [number of src] [array radius] [optitrack ip] [multicast address] [optitrack port] [ssr end message]

`opti_ssr_demo_localwfs.`**`demo`**(*ssr_ip='localhost'*, *ssr_port=4711*, *ssr2_port=4712*, *opti_unicast_ip=None*, *opti_multicast_ip='239.255.42.99'*, *opti_port=1511*, *ssr_end_message='\x00'*)

A demo function to track the listener position.

Parameters

- **ssr_ip** (*str, optional*) – IP of the server running the SSR.
- **ssr_port** (*int, optional*) – Port of the first SSR's Network Interface. By default, port 4711.
- **ssr2_port** (*int, optional*) – Port of the second SSR's Network Interface. By default, port 4712.
- **opti_unicast_ip** (*str, optional*) – IP of the Motive software to establish a unicast connection to. By default, no unicast connection is established.
- **opti_multicast_ip** (*str, optional*) – Multicast address to connect to.
- **opti_port** (*int, optional*) – Port of the Motive network interface.
- **ssr_end_message** (*str, optional*) – Symbol to terminate the XML message sent to SSR. By default, a binary zero.

3.2 Headtracker

A python module for demonstrating head orientation tracking for binaural synthesis.

Usage: python opti_ssr_demo.py [SSR_IP] [SSR_port] [optitrack ip] [multicast address] [optitrack port] [end_message]

opti_ssr_demo_headtracker.**demo**(*ssr_ip='localhost'*, *ssr_port=4711*, *opti_unicast_ip=None*, *opti_multicast_ip='239.255.42.99'*, *opti_port=1511*, *ssr_end_message='\x00'*)

A demo function to track the head orientation.

Parameters

- **ssr_ip** (*str, optional*) – IP of the server running the SSR.
- **ssr_port** (*int, optional*) – Port of SSR Network Interface. By default, port 4711.
- **opti_unicast_ip** (*str, optional*) – IP of the Motive software to establish a unicast connection to. By default, no unicast connection is established.
- **opti_multicast_ip** (*str, optional*) – Multicast address to connect to.
- **opti_port** (*int, optional*) – Port of the Motive network interface.
- **ssr_end_message** (*str, optional*) – Symbol to terminate the XML message sent to SSR. By default, a binary zero.

3.3 API Documentation

3.3.1 Optitrack Client

A python module to connect to Optitrack optical tracking system and receive data from it.

class opti_ssr.opti_client.**OptiTrackClient**(*unicast_ip=None*, *multicast_ip='239.255.42.99'*, *port=1511*, *natnet_version=(3, 0, 0, 0)*)

Connect to Optitrack systems and Motive software and receive data, including rigid body position and orientation, from it. By default, it connects to Optitrack software Motive on the same machine.

unicast_ip
str, optional – IP of the Motive software to establish a unicast connection to. By default, no unicast connection is established.

multicast_ip
str, optional – Multicast address to connect to.

port
int, optional – Port of the Motive network interface.

natnet_version
tuple, optional – Version number of the NatNetSDK to use.

get_packet_data(*packet_types=[<class 'opti_ssr.optirx.SenderData'>, <class 'opti_ssr.optirx.ModelDefs'>, <class 'opti_ssr.optirx.FrameOfData'>]*)
Receive desired packet data.

based on optirx-demo.py source: https://bitbucket.org/astanin/python-optirx

Parameters **packet_types** (*list, optional*) – Types of the packets to be returned.

Returns **packet** (*list*) – Received packets of desired type.

get_rigid_body(*rb_id=0*)
Receive rigid body position, orientation and time data.

Parameters **rb_id** (*int, optional*) – ID of the rigid body to receive data from.

Returns

- **position** (*numpy array*) – Rigid body position packet data of the desired rigid body. Consists of x, y, z coordinates of Motive's coordinate system.
- **orientation** (*list*) – List of rigid body orientation data in quaternion representation.
- **time_data** (*list*) – List of time data consisting of frame mumber, timestamp and latency packet data.

3.3.2 SSR Client

A python module for controlling the SoundScape Renderer.

class opti_ssr.ssr_client.**SSRClient**(*ip='localhost'*, *port=4711*, *end_message='x00'*)
Establish a TCP/IP4 network connection and send XML messages to communicate with a specific instance of the SoundScape Renderer.

ip
str, optional – IP of the server running thr SSR. By default, it connects to localhost.

port
int, optional – Port of SSR Network Interface. By default, port = 4711.

end_message
str, optional – Symbol to terminate the XML messages send to SSR. By default, a binary zero.

load_scene(*path*)
Load a scene from a specified location on the machine running the SSR.

recv_ssr_returns()
Receive messages returned by the SSR.

set_ref_offset_orientation(*alpha*)
Set reference offset orientation in degrees (zero in positive x-direction).

set_ref_offset_position(*x*, *y*)
Set reference offset position in meters.

set_ref_orientation(*alpha*)
Set reference orientation in degrees (zero in positive x-direction).

set_ref_position(*x*, *y*)
Set reference position in meters.

set_src_orientation(*src_id*, *alpha*)
Change orientation of an existing source in degrees (zero in positive x-direction).

set_src_position(*src_id*, *x*, *y*)
Change name and position of an existing source.

set_transport_state(*state*)
Set a specific transport state, namely start, stop or rewind to play, pause or rewind all audio tracks of the loaded scene respectively.

src_creation(*src_id*)
Define a new source.

3.3.3 Bridges Module

A python module that provides tools for position and orientation tracking inside the SoundScape Renderer (SSR) using the OptiTrack optical tracking system.

This module contains the abstract class _Bridge and its subclasses. _Bridge is a thread class that includes the actual sequence, whereas the subclasses implement different applications of a connection between the SSR and a tracking system. In any subclass of _Bridge the functions to receive and send data need to be defined according to the desired application.

class opti_ssr.bridges.**HeadTracker**(*optitrack*, *ssr*, *rb_id=0*, *angle=1*, **args*, ***kwargs*)
Bases: *opti_ssr.bridges._Bridge*

A class for using the OptiTrack system as a head tracker for the SSR.

optitrack
class object – Object of class OptiTrackClient.

ssr
class object – Object of class SSRClient.

rb_id
int, optional – ID of the rigid body to receive data from.

angle
int, optional – angle which is used for head rotation

- +1 - positive yaw
- -1 - negative yaw
- +2 - positive pitch
- -2 - negative pitch
- +3 - positive roll
- -3 - negative roll

calibrate()
Use current position and orientation of head tracker to set the origin and orientation of the world coordinate system.

class opti_ssr.bridges.**LocalWFS**(*optitrack*, *ssr*, *ssr_virt_repr*, *rb_id=0*, **args*, ***kwargs*)
Bases: *opti_ssr.bridges._Bridge*

A class for using the OptiTrack system to track the listener position in the SSR for local sound field synthesis.

The first SSR instance (ssr) shifts the reference position and the reference offset position of the real reproduction setup, such that it emulates the movement a virtual source array placed around the listener.

The second SSR instance (ssr_virt_repr) shifts the reference position of aforementioned point sources as the virtual reproduction setup in relation to the real sources based on audio files.

optitrack
class object – Object of the class OptiTrackClient.

ssr
class object – First SSR instance as object of class SSRClient.

ssr_virt_repr
class object – Second SSR instance as object of the class SSRClient.

rb_id
int, optional – ID of the rigid body to receive data from.

_create_virtual_sources()
Create a specified amount of new sources via network connection to the SSR.

_receive()
Get position data of rigid body from OptiTrack system

Returns center (*list*) – Rigid body position data. Consists of x, y, z coordinates of Motive's coordinate system.

_send(*center*)
Send reference position data to both SSR instance.

class opti_ssr.bridges.**_Bridge**(*optitrack*, *ssr*, *data_limit=500*, *timeout=0.01*, **args*, ***kwargs*)
Bases: threading.Thread

An abstract class which implements a threading approach to receive and send data. To implement the functionality to send and receive the desired data, subclasses need to define the functions _receive and _send.

Note: The returns of _receive have to be the input of _send.

---

clear_data()
Clears buffer

get_last_data(*num=None*)
Returns a list of data received from the OptiTrack system.

run()
Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

3.3.4 Optirx Module

A pure Python library to receive motion capture data from OptiTrack. The version 1.10 of this optirx library is included here with only minor changes from the original source.

See original source: https://bitbucket.org/astanin/python-optirx

CHAPTER 4

Index

- genindex
- modindex

CHAPTER 5

Version History

Version 0.1.4 (2018-06-13):

- Added option to change ref_offset_orientation in ssr_client.
- Remove necessity of knowing the geometry of virtual loudspeaker distribution for LocalWFS
- Add parameter to define, which euler angle of the RigidBody is selected for the orientation of the HeadTracker

Version 0.1.3 (2017-03-22):

- Added option to change ref_position_offset in ssr_client.
- Added single ref_orientation_shift for local WFS.
- Added continuous ref_position_offset shift to local WFS second instance.

Version 0.1.2 (2017-02-23):

- Fixed wrong angle used for head orientation tracking due to the fix in version 0.1.1.
- Added version history in documentation.

Version 0.1.1 (2017-02-23):

- Fixed a bug in orientation representation.

Version 0.1.0 (2017-02-14): Initial release.

Python Module Index

o

Index

Symbols

A

C

D

E

G

H

I

L

M

N

O

P

R

S

U