Avpy Documentation

Release 0.1.3

sydh

May 01, 2016

Contents

CHAPTER 1

Overview

Avpy is a ctypes (python) binding for libav and ffmpeg.

Examples Code examples.

Frequently Asked Questions Some questions that come up often.

avpy The complete API documentation, organized by classes and functions.

CHAPTER 2

Getting Help

If you're having trouble or have questions, you can write me an email (sydhds __at__ gmail __dot__ com) or create an issue in the bug tracker (see Issues) and mark it as an ehancement or a proposal.

CHAPTER 3

Issues

All issues should reported at the Avpy bug tracker.

CHAPTER 4

Changes

See the Changelog for a full list of changes.

CHAPTER 5

Contributions

Fork the git repo and submit a pull request.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

6.1 Examples

The examples in this section are intended to give in depth overviews of how to accomplish specific tasks in Python.

Unless otherwise noted, all examples assume that the Avpy module can be imported.

See also:

see 'examples' folder in avpy repository

6.1.1 First Steps

This example covers your first steps with Avpy.

Importing

Let's start with printing some media information:

```
from avpy import Media
# print media info (streams, codecs...)
m = Media('test.avi')
print m.info()
```

Going further

- Try the next example in Examples
- Explore the API documentation
- *Media*

6.2 Frequently Asked Questions

Contents

6.2.1 Libav and FFmpeg version support?

- Libav 0.8, 9, 10 and 11 (all patch version supported)
- FFmpeg 1.2 (all patch version supported)

Note that support for FFmpeg 2.2, 2.4 and 2.5 will be added later.

6.2.2 Does Avpy support Python 3?

Yes for Python 3.2+ (pypy is supported as well). Please report if something is broken for Python3.

6.2.3 Is the high level API stable?

Short answer: Almost but not yet.

Long answer:

As the api provides support for decoding and encoding, it should remain almost stable. Some additional code will be provided to abstract as much as possible any 'ctypes code' and to support new features (filtering, audio resampling, seeking ...).

6.2.4 Does Avpy only provide a high level API?

Short answer: A low level API is available but its usage is not recommended.

Long answer:

A low level API is available as ctypes functions that directly map the underlying C functions.

Note that this API varies from library (ffmpeg or libav) and from version (libav0.8 and libav 9). So while the high level API take cares of theses differences, the low API does not.

Available functions are declared in avpy/version/av{VERSION}.py. Please report if some functions are missing.

6.3 avpy

6.3.1 avpy package

Submodules

avpy.av module

avpy.avMedia module

class avpy.avMedia.**Media**(*mediaName*, *mode='r'*, *quiet=True*)
Bases: object

__init__(*mediaName*, *mode='r'*, *quiet=True*)
Initialize a media object for decoding or encoding

Parameters

- **mediaName** – media to open
- **mode** (*str*) – 'r' (decoding) or 'w' (encoding)
- **quiet** (*bool*) – turn on/off libav or ffmpeg warnings

addResampler(*streamIndex*, *inAudio*, *outAudio*)
Add an audio resampler

A resampler is responsible for:

•resampling audio (frequency, layout)

•converting output data format (ie s16p -> s16)

Parameters

- **inAudio** (*dict*) – audio input info
- **outAudio** (*dict*) – audio output info

Note: each audio info requires the following fields:

•sampleRate: sample rate

•sampleFmt: sample format name (ie s16)

•layout (optional): channel layout (ie. mono, stereo...)

•channels (optional): channel count

if 'layout' is not specified, its value is guessed from the channel count

---

addScaler(*streamIndex*, *width*, *height*, *pixelFormat='rgb24'*, *scaling='bilinear'*)
Add a scaler

A scaler is responsible for:

•scaling a video

•converting output data format (ie yuv420p to rgb24)

Parameters

- **streamIndex** (*int*) – stream index
- **with** – new stream width
- **height** (*int*) – new stream height
- **pixelFormat** (*str*) – output pixel format
- **scaling** (*str*) – scaling algorithm

See also:

•https://en.wikipedia.org/wiki/Image_scaling

Note: Available scaling algorithm

•fast_bilinear

•bilinear

•bicubic

•area

•bicubiclin (bicubic for luma, bilinear for chroma)

•gauss

•sinc

•lanczos

•spline

---

addStream(*streamType*, *streamInfo*)

Add a stream

After opening a media for encoding (writing), a stream of desired type (video or audio) have to be added.

Parameters

- **streamType** (*str*) – video or audio stream
- **streamInfo** (*dict*) – stream parameters

Supported stream parameters:

•video:

- width, height: video size
- pixelFormat: pixel format name (ie. rgb24, yuv420p ...)
- codec: video codec name (ie. mp4), auto is a valid value
- timeBase: as a tuple (ie (1, 25) for 25 fps)
- bitRate: average bit rate (ie 64000 for 64kbits/s)

•audio:

- sampleRate: sample frequency (ie. 44100 for 44.1 kHz)
- bitRate: average bit rate (see video bit rate)
- channels channel count (ie. usually 2)

- codec: audio codec name, auto is a valid value
- sampleFmt: sample format (ie flt for float)

More parameters are documented here: Encoding

Note:

•For an image, timeBase and bitRate parameters are ignored

•More parameters will be supported in the near futur

•Subtitle streams are not yet supported

Note: use *codecInfo()* to query codec caps

---

audioPacket()

Get an audio packet ready for encoding purpose

Initialize and allocate data for an audio packet. Data format will depend to the previously added stream.

Returns video packet

Return type *Packet*

Raises RuntimeError if data could not be allocated

info()

Get media information

Returns dict with the following fields: name, metadata, stream, duration

Return type dict

•duration: media duration in seconds

•name: media filename

•stream: list of stream info (dict)

Note: each stream info will contains the following fields:

•all:

- codec: codec short name
- type: video, audio or subtitle

•video:

- widht, height: video size
- fps: as a tuple of 3 values (num, den, ticks)
- pixelFormat: pixel format name (ie. rgb24, yuv420p ...)

•audio:

- sampleRate: sample rate
- channels: channel count
- sampleFmt: sample format name (ie. s16 ...)

- sampleFmtId: sample format id (internal use)
- frameSize: frame size
- bytesPerSample: bytes for sample format (ie. 2 for s16)

•**subtitle:**

- subtitle header: header string

See also:

writeHeader()

metadata()
Get media metadata

Returns a dict with key, value = metadata key, metadata value

---

Note: method is also called by *info()*

See also:

writeHeader()

next()
Iterate over Media

Return type *Packet*

static **open**(**args*, ***kwds*)
Open a media file (writing only)

Parameters

- **mediaName** (*str*) – media name
- **mode** (*str*) – open media mode (ignored)
- **buffering** (*int*) – buffering (ignored)
- **metaData** (*dict*) – meta data dict (optional)
- **streamsInfo** (*dict*) – stream(s) info

videoPacket()
Get a video packet ready for encoding purpose

Initialize and allocate data for a video packet. Data format will depend to the previously added stream.

Returns video packet

Return type *Packet*

write(*packet*, *pts*, *mediaType='video'*)
Write packet to media

Parameters **packet** (*Packet* or array.array or numpy.array) – packet to encode and add to media

writeHeader(*metaData=None*)
Write media header

Write media header. This method have to be called before any call to *write()*

Parameters **metaData** (*dict*) – media metaData (ie. artist, year ...)

Note: see http://multimedia.cx/eggs/supplying-ffmpeg-with-metadata/ for available metadata per container

writeTrailer()
Write media trailer

Write media trailer. Call this method just before closing or deleting media.

class avpy.avMedia.**Packet**(*formatCtx*)
Bases: object

Media data container

When decoding a media, a packet object will be returned and might be decoded later.

When encoding, a packet is retrieved then written.

__init__(*formatCtx*)

addResampler(*streamIndex*)
Add an audio resampler

Note: called by *Media.addResampler()*

decode()
Decode data

streamIndex()

avpy.avMedia.**avError**(*res*)
Return an error message from an error code

The libav or ffmpeg functions can return an error code, this function will do its best to translate it to an human readable error message.

Parameters **res** (*int*) – error code

Returns error message

Return type str

Note: if error is unknown, return 'Unknown error code %d'

avpy.avMedia.**codecInfo**(*name*, *decode=True*)
Retrieve specific codec information

Parameters

- **name** (*bool*) – codec name
- **decode** – codec decoder info. Set decode to False to get codec encoder info.

Returns codec information

Return type dict

Raises ValueError if codec name is unknown

Note: codec information keys:

- •name
- •longname
- •type: video, audio or subtitle
- •thread: codec thread capacity
- •autoThread: auto thread support
- •framerates: supported frame rates
- •samplerates: supported sample rates
- •pixFmts: supported pixel formats
- •profiles: supported encoding profiles
- •sampleFmts: supported sample formats

avpy.avMedia.**codecs**()
Get all supported codecs

Returns a dict with 3 keys (audio, video and subtitle). For each key, the value is a dict with 2 keys (encoding and decoding).

Return type dict

avpy.avMedia.**formats**()
Get all supported formats

Returns a dict with 2 keys: muxing and demuxing

Return type dict

avpy.avMedia.**versions**()
Return version & config & license of C libav or ffmpeg libs

Returns dict with keys: version, configuration, license, path

Return type dict

Module contents

6.3.2 Encoding

Additional parameters

- **gopSize:**
  - the number of pictures in a group of pictures, or 0 for intra_only
  - http://en.wikipedia.org/wiki/Group_of_pictures
- **maxBFrames:**
  - maximum number of B-frames between non-B-frames
  - https://en.wikipedia.org/wiki/Inter_frame
- **mbDecision:**

  - macroblock decision mode
  - https://en.wikipedia.org/wiki/Macroblock

6.4 Changelog

6.4.1 Changes in Version 0.1.2

- Audio resampling support
- FFmpeg 2.5, 2.6, 2.7 and 2.8 support

6.4.2 Changes in Version 0.1.1

- Add scaler arguments (pixel format, scaling algo)
- Python3 fixes

6.4.3 Changes in Version 0.1.0

- Initial release

a

Symbols

A

C

D

F

I

M

N

O

P

S

V

W