thumbor-video-engine

Release 1.2.2

Aug 14, 2021

Contents

thumbor-video-engine provides a thumbor engine that can read, crop, and transcode audio-less video files. It supports input and output of animated GIF, animated WebP, WebM (VP9) video, and MP4 (default H.264, but HEVC is also supported).

CHAPTER 1

Installation

```
pip install thumbor-video-engine
```

Go to GitHub if you need to download or install from source, or to report any issues.

CHAPTER 2

Setup

In your thumbor configuration file, change the `ENGINE` setting to `'thumbor_video_engine.engines.video'` to enable video support. This will allow thumbor to support video files in addition to whatever image formats it already supports. If the file passed to thumbor is an image, it will use the Engine specified by the configuration setting `IMAGING_ENGINE` (which defaults to `'thumbor.engines.pil'`).

To enable transcoding between formats, add `'thumbor_video_engine.filters.format'` to your `FILTERS` setting. If `'thumbor.filters.format'` is already present, replace it with the filter from this package.

```
ENGINE = 'thumbor_video_engine.engines.video'
FILTERS = [
    'thumbor_video_engine.filters.format',
    'thumbor_video_engine.filters.still',
]
```

To enable automatic transcoding to animated gifs to webp, you can set `FFMPEG_GIF_AUTO_WEBP` to `True`. To use this feature you **cannot** set `USE_GIFSICLE_ENGINE` to `True`; this causes thumbor to bypass the custom `ENGINE` altogether. If you still want gifsicle to handle animated gifs you should set `FFMPEG_USE_GIFSICLE_ENGINE` to `True` and set `GIF_ENGINE` to `"thumbor_video_engine.engines.gif"`.

You can also tell thumbor-video-engine to auto-transcode animated gifs to h264 or h265 when an HTTP request has `video/*` in its `Accept` header by enabling `FFMPEG_GIF_AUTO_H264` or `FMPEG_GIF_AUTO_H265`, respectively. If it possible to use these settings along with `FFMPEG_GIF_AUTO_WEBP`. If a request were to send an `Accept` header for both webp and video (e.g. `image/webp, video/*`) the engine would return an h264/h265 mp4. However, at the time this documentation is written there aren't any major browsers that accept both webp *and* videos for images.

If your thumbor application is behind a CDN or caching proxy and you're using any of the automatic gif transcoding options, you will need to set `APP_CLASS` to `"thumbor_video_engine.app.ThumborServiceApp"` to ensure that thumbor returns `Vary: Accept` when appropriate.

If you want to use auto-mp4 gif conversion with result storage, you will need to set your `RESULT_STORAGE` to one that stores the auto-converted mp4 videos separately from auto-webp or non-auto-converted gifs. This module provides a subclassed version of `thumbor.result_storages.file_storage` in `thumbor_video_engine.result_storages.file_storage`. For those using the s3 result_storage class from tc_aws you can set

RESULT_STORAGE to thumbor_video_engine.result_storages.s3_storage to enable storage of gifs autoconverted to mp4.

```
ENGINE = 'thumbor_video_engine.engines.video'
GIF_ENGINE = 'thumbor_video_engine.engines.gif'
FFMPEG_USE_GIFSICLE_ENGINE = True
FFMPEG_GIF_AUTO_WEBP = True
FFMPEG_GIF_AUTO_H265 = True
APP_CLASS = 'thumbor_video_engine.app.ThumborServiceApp'
RESULT_STORAGE = 'thumbor_video_engine.result_storages.file_storage'
```

2.1 Contents

2.1.1 Configuration

All configuration values default to None unless otherwise specified. For FFmpeg codec settings, a None value means that it uses ffmpeg's own default value for when the flag is unspecified.

Here are some general resources on ffmpeg's encoding flags, and how to choose encoding settings that fit your use-case.

- Understanding Rate Control Modes
- Google: VP9 Overview
- FFmpeg H.264 Encoding Guide
- FFmpeg H.265 Encoding Guide
- FFmpeg VP9 Encoding Guide
- FFmpeg codecs documentation

General

IMAGE_ENGINE

The engine to use for non-video files. It defaults to 'thumbor.engines.pil', which is thumbor's default value for ENGINE

FFMPEG_ENGINE

The engine to use for video files. It defaults to 'thumbor_video_engine.engines.ffmpeg'.

GIFSICLE_PATH

The path to the gifsicle binary. It defaults to None, in which case it looks for gifsicle in PATH. This is only used if GIF_ENGINE is set to 'thumbor_video_engines.engines.gif'. As of version 6.7.0, thumbor does not support configuring this value.

GIFSICLE_ARGS

A list of additional args to pass to gifsicle. This is only used if `GIF_ENGINE` is set to `'thumbor_video_engines.engines.gif'`.

FFMPEG_USE_GIFSICLE_ENGINE

Equivalent to USE_GIFSICLE_ENGINE, but for the FFmpeg engine. It defaults to `False`. If `True`, it will perform any image operations on animated gifs (e.g. cropping and resizing) using gifsicle (by way of `GIF_ENGINE`).

FFMPEG_HANDLE_ANIMATED_GIF

Whether to process animated gifs with the FFmpeg engine. It defaults to `True`.

FFMPEG_GIF_AUTO_WEBP

Specifies whether animated WebP format should be used automatically if the source image is an animated gif and the request accepts it (via Accept header). It defaults to `True`, but only works when `AUTO_WEBP` is also enabled.

FFMPEG_GIF_AUTO_H264

Specifies whether H264 format should be used automatically if the source image is an animated gif and the request accepts it (via `Accept: video/*`). It defaults to `False`.

FFMPEG_GIF_AUTO_H265

Specifies whether H265 format should be used automatically if the source image is an animated gif and the request accepts it (via `Accept: video/*`). It defaults to `False`.

FFPROBE_PATH

Path for the ffprobe binary. It defaults to `'/usr/local/bin/ffprobe'`.

H.264 (MP4)

FFMPEG_H264_TWO_PASS

Whether to use two-pass encoding for h264 in FFmpeg. Default `False`.

FFMPEG_H264_CRF

-crf. The constant quality to use by FFmpeg for h264 encoding.

FFMPEG_H264_VBR

-b:v: The average bitrate to be used by FFmpeg for h264 encoding.

FFMPEG_H264_MINRATE

-minrate: minimum bound for bitrate.

FFMPEG_H264_MAXRATE

-maxrate: maximum bound for bitrate.

FFMPEG_H264_BUFSIZE

-bufsize: The rate control buffer. Used to determine the range across which the requested average bitrate and min/max should be enforced.

FFMPEG_H264_PRESET

-preset. A collection of options that will provide a certain encoding speed to compression ratio. A slower preset will provide better compression (compression is quality per filesize). This means that, for example, if you target a certain file size or constant bit rate, you will achieve better quality with a slower preset. Similarly, for constant quality encoding, you will simply save bitrate by choosing a slower preset.

Use the slowest preset that you have patience for. The available presets in descending order of speed are: *ultrafast*, *superfast*, *veryfast*, *faster*, *fast*, *medium* (default), *slow*, *slower*, *veryslow*

FFMPEG_H264_PROFILE

-profile: Determines h264 compatibility version.

FFMPEG_H264_LEVEL

-level: Controls h264 feature set, which affects device compatibility.

FFMPEG_H264_TUNE

-tune: Change settings based upon the specifics of your input

FFMPEG_H264_QMIN

-qmin: Set the minimum video quantizer scale.

FFMPEG_H264_QMAX

-qmax: Set the maximum video quantizer scale.

H.265 (aka HEVC)

FFmpeg H.265 Encoding Guide

FFMPEG_H265_TWO_PASS

Whether to use two-pass encoding for h265 encoding. Default `False`.

FFMPEG_H265_PRESET

-preset. A collection of options that will provide a certain encoding speed to compression ratio. Same values as h264

FFMPEG_H265_LEVEL

-level: Controls h265 feature set, which affects device compatibility.

FFMPEG_H265_MAXRATE

The –vbv-maxrate flag passed to FFmpeg for h265 encoding.

FFMPEG_H265_BUFSIZE

The –vbv-bufsize flag passed to libx265.

FFMPEG_H265_CRF_MIN

The –crf-min flag passed to libx265.

FFMPEG_H265_CRF_MAX

The –crf-max flag passed to libx265.

FFMPEG_H265_PROFILE

-profile: Determines h265 compatibility version.

FFMPEG_H265_TUNE

-tune: Change settings based upon the specifics of your input. Same as h264.

FFMPEG_H265_CRF

-crf: the constant quality to use by FFmpeg for h264 encoding.

FFMPEG_H265_VBR

-b:v: The average bitrate to be used by FFmpeg for h265 encoding.

VP9 (WebM)

FFMPEG_VP9_TWO_PASS

Whether to use two-pass encoding for VP9 in FFmpeg. Default `False`.

FFMPEG_VP9_VBR

-b:v. The average bitrate to be used by FFmpeg for VP9 encoding.

FFMPEG_VP9_LOSSLESS

-lossless. Whether to enable lossless encoding for VP9. Default `False`.

FFMPEG_VP9_DEADLINE

-deadline: can be set to:

good the default and recommended for most applications.

best recommended if you have lots of time and want the best compression efficiency.

realtime recommended for live / fast encoding.

FFMPEG_VP9_CRF

-crf. The constant quality to use by FFmpeg for VP9 encoding.

FFMPEG_VP9_CPU_USED

-cpu-used: Affects compilation speed and quality trade-off

FFMPEG_VP9_ROW_MT

-row-mt. Whether to enable row-based multithreading for VP9 encoding.

FFMPEG_VP9_MINRATE

-minrate: minimum bound for bitrate.

FFMPEG_VP9_MAXRATE

-maxrate: maximum bound for bitrate.

Animated WebP

FFMPEG_WEBP_LOSSLESS

-lossless: enables/disables use of lossless mode. libwebp default is `False`.

FFMPEG_WEBP_COMPRESSION_LEVEL

-compression_level: range 0-6, default 4. Higher values give better quality but slower speed. For lossless, it controls the size/speed trade-off.

FFMPEG_WEBP_QSCALE

-qscale: For lossy encoding, controls quality 0 to 100. For lossless, controls cpu and time spent compressing. libwebp built-in default 75.

FFMPEG_WEBP_PRESET

-preset Configuration preset. Consult FFmpeg libwebp codec documentation for more information.

Example Configuration

```
ENGINE = 'thumbor_video_engine.engines.video'
FFMPEG_USE_GIFSICLE_ENGINE = True
FFMPEG_PATH = '/usr/bin/ffmpeg'
FFPROBE_PATH = '/usr/bin/ffprobe'
FFMPEG_H264_MAXRATE = '1200k'
FFMPEG_H264_BUFSIZE = '2400k'
FFMPEG_H264_CRF = 24
FFMPEG_H265_MAXRATE = '1500'
FFMPEG_H265_BUFSIZE = '3000'
FFMPEG_H265_CRF = 28
FFMPEG_VP9_VBR = '2M'
FFMPEG_VP9_CRF = 30
FFMPEG_VP9_MINRATE = '1500k'
FFMPEG_VP9_MAXRATE = '2500k'
FFMPEG_VP9_CPU_USED = 4
FFMPEG_VP9_ROW_MT = True
FFMPEG_WEBP_COMPRESSION_LEVEL = 3
FFMPEG_WEBP_QSCALE = 80
```

2.1.2 Filters

format(*image-or-video-format*)

http://thumbor-server/filters:format(webm)/some/video.mp4

This filter specifies the output format of the image or video. The argument must be one of: "png", "jpeg", "gif", "webp", "h264", "h265", or "vp9". It also accepts aliases of "mp4" (for h264), "webm" (for vp9), and "hevc" (for h265).

still()

http://thumbor-server/1280x800/filters:still()/some/video.mp4

This filter returns the first frame of a video as a still image. The resulting image is a jpeg by default, but this can be overridden with the format filter.

lossless()

http://thumbor-server/filters:lossless:format(webm)/some/video.mp4

This filter enables lossless encoding, if the output format supports it (currently only animated WebP and VP9 (aka WebM)).

2.1.3 Changelog

1.2.2 (Aug 14, 2021)

- Support source videos in quicktime/mov format. Fixes #9.

1.2.1 (Jul 20, 2021)

- Ensure that `Vary: Accept` header is returned for requests to animated gifs that can be auto-converted to other formats when that image is returned from result storage.

1.2.0 (Jul 19, 2021)

- Added an `APP_CLASS "thumbor_video_engine.app.ThumborServiceApp"` that ensures appropriate `Vary: Accept` header is returned for requests that automatically convert animated gifs based on Accept headers.
- Added settings `FFMPEG_GIF_AUTO_H264` and `FFMPEG_GIF_AUTO_H265` that enable auto-conversion of animated gifs to H264 or H265 (respectively) when `video/*` is present in a request's `Accept` header.
- Added custom result storage classes that ensures animated gifs auto-converted to mp4 are stored distinctly from animated gifs or auto-webp images.

1.1.1 (Feb 25, 2021)

- Added `GIFSICLE_ARGS` setting, which allows customization of arguments passed to the gifsicle binary.

1.1.0 (May 29, 2020)*

- Added support for python 3 and thumbor 7 alpha
- Fixed: conversion to animated webp now retains alpha transparency (from gif or webp)

1.0.2 (Jan 11, 2020)

- Feature: Support auto conversion of animated gif to animated webp if `AUTO_WEBP` is `True`.
- Fixed: Ensure that all mp4s, regardless of `ftyp` box size, are recognized as such.

1.0.1 (Jan 8, 2020)

- Fixed: Addressed an issue where frames would sometimes get dropped when transcoding animated webp files.

1.0.0 (Jan 2, 2020)

- Initial release

2.2 License

This code is licensed under the MIT License. View the `LICENSE` file under the root directory for complete license and copyright information.

2.3 Indices and tables

- genindex
- modindex
- search