Package ‘AV1R’

May 6, 2026

Type Package

Title 'AV1' Video Encoding for Biological Microscopy Data

Version 0.1.3

Description Converts legacy microscopy video formats (H.264/H.265, AVI/MJPEG, TIFF stacks) to the modern 'AV1' codec with minimal quality loss. Typical use cases include compressing large TIFF stacks from confocal microscopy and time-lapse experiments from hundreds of gigabytes to manageable sizes, re-encoding MP4 files exported from 'CellProfiler', 'ImageJ'/'Fiji', and microscope software with approximately 2x better compression at the same visual quality, and converting legacy AVI (MJPEG) and H.265 recordings to a single patent-free format suited for long-term archival. Automatically selects the best available backend: GPU hardware acceleration via 'Vulkan' 'VK_KHR_VIDEO_ENCODE_AV1' or 'VAAPI' (tested on AMD RDNA4; bundled headers, builds with any 'Vulkan' SDK >= 1.3.275), with automatic fallback to CPU encoding through 'FFmpeg' and 'SVT-AV1'. User controls quality via a single CRF parameter; each backend adapts automatically (CPU and Vulkan use CRF directly, VAAPI targets 55 percent of input bitrate). TIFF stacks use near-lossless CRF 5 by default, with optional proportional scaling via tiff_scale (multiplier or bounding box, aspect ratio always preserved). Small frames are automatically scaled up to meet hardware encoder minimums. Audio tracks are preserved automatically. Provides a simple R API for batch conversion of entire experiment folders.

Depends R (>= 4.1.0)

License MIT + file LICENSE

URL https://github.com/Zabis13/AV1R

BugReports https://github.com/Zabis13/AV1R/issues

Encoding UTF-8

SystemRequirements C++17, GNU make, FFmpeg (>= 4.4) with libavcodec/libavformat, libvulkan-dev (optional, for GPU encoding on Linux), Vulkan SDK (optional, for GPU encoding on Windows)

Suggests magick, testthat (>= 3.0.0)

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation yes

Author Yuri Baramykov [aut, cre]

Maintainer Yuri Baramykov <lbsbmsu@mail.ru>

Repository CRAN

Date/Publication 2026-03-23 08:40:40 UTC

Contents

AV1R-package 2
av1r_options 3
av1r_status 4
convert_folder 5
convert_to_av1 6
detect_backend 7
measure_ssim 7
read_tiff_stack 8
vulkan_available 8
vulkan_devices 9

Index **10**

`AV1R-package` *AV1R: 'AV1' Video Encoding for Biological Microscopy Data*

Description

Converts legacy microscopy video formats (H.264/H.265, AVI/MJPEG, TIFF stacks) to the modern 'AV1' codec with minimal quality loss. Typical use cases include compressing large TIFF stacks from confocal microscopy and time-lapse experiments from hundreds of gigabytes to manageable sizes, re-encoding MP4 files exported from 'CellProfiler', 'ImageJ'/'Fiji', and microscope software with approximately 2x better compression at the same visual quality, and converting legacy AVI (MJPEG) and H.265 recordings to a single patent-free format suited for long-term archival. Automatically selects the best available backend: GPU hardware acceleration via 'Vulkan' 'VK_KHR_VIDEO_ENCODE_AV1' or 'VAAPI' (tested on AMD RDNA4; bundled headers, builds with any 'Vulkan' SDK >= 1.3.275), with automatic fallback to CPU encoding through 'FFmpeg' and 'SVT-AV1'. User controls quality via a single CRF parameter; each backend adapts automatically (CPU and Vulkan use CRF directly, VAAPI targets 55 percent of input bitrate). TIFF stacks use near-lossless CRF 5 by default, with optional proportional scaling via tiff_scale (multiplier or bounding box, aspect ratio always preserved). Small frames are automatically scaled up to meet hardware encoder minimums. Audio tracks are preserved automatically. Provides a simple R API for batch conversion of entire experiment folders.

Author(s)

Maintainer: Yuri Baramykov <lbsbmsu@mail.ru>

See Also

Useful links:

- https://github.com/Zabis13/AV1R
- Report bugs at https://github.com/Zabis13/AV1R/issues

---

av1r_options *AV1R encoding options*

Description

AV1R encoding options

Usage

```
av1r_options(
  crf = 28L,
  preset = 8L,
  threads = 0L,
  bitrate = NULL,
  tiff_crf = 5L,
  tiff_scale = NULL,
  verbose = TRUE,
  backend = "auto"
)
```

Arguments

`crf`	Constant Rate Factor: 0 (best) to 63 (worst). Default 28. Lower = better quality, larger file. Used as fallback when `bitrate` is `NULL` and input bitrate cannot be detected.
`preset`	Encoding speed preset: 0 (slowest/best) to 13 (fastest). Default 8 (good balance for microscopy batch jobs).
`threads`	Number of CPU threads. 0 = auto-detect.
`bitrate`	Target video bitrate in kbps (e.g. `3000` for 3 Mbps). `NULL` (default) = auto-detect from input (55% of source bitrate for VAAPI, CRF for CPU).
`tiff_crf`	CRF for TIFF/image sequence input. Default 5 (near-lossless, suitable for microscopy data). Overrides `crf` when input is TIFF or image sequence.
`tiff_scale`	Scale factor or target size for TIFF/image input. `NULL` (default) = no scaling (only auto-scale to meet hardware minimum). A single number = multiplier (e.g. `2` doubles both dimensions). A vector `c(width, height)` = target bounding box (image is scaled proportionally to fit inside, preserving aspect ratio).

verbose	Logical. If TRUE (default), print informational messages about encoding decisions (e.g. TIFF CRF override).
backend	"auto" (best available: vaapi > vulkan > cpu), "vaapi" (VAAPI AV1, AMD/Intel), "vulkan" (Vulkan AV1), or "cpu" (SVT-AV1 via ffmpeg).

Value

A named list of encoding parameters.

Examples

```
# Default options (auto-detect backend and bitrate)
av1r_options()

# High-quality lossless-ish for publication figures
av1r_options(crf = 15, preset = 4)

# Explicit bitrate (GPU VAAPI)
av1r_options(bitrate = 2000, backend = "vaapi")

# Fast batch conversion of large TIFF stacks
av1r_options(crf = 32, preset = 12, threads = 16)
```

---

av1r_status *Show AV1R backend status*

Description

Prints the active encoding backend and available GPU/CPU capabilities.

Usage

```
av1r_status()
```

Value

Invisibly returns the active backend string.

convert_folder *Convert all videos in a folder to AV1*

Description

Finds all supported video files in `input_dir` and converts them to AV1. Output files are written to `output_dir` with the same base name and `.mp4` extension.

Usage

```
convert_folder(
  input_dir,
  output_dir = input_dir,
  options = av1r_options(),
 ext = c("mp4", "avi", "mkv", "mov", "flv", "mpg", "mpeg", "webm", "tif", "tiff"),
  skip_existing = TRUE,
  recursive = FALSE,
  max_depth = 5L
)
```

Arguments

`input_dir`	Path to folder with input files.
`output_dir`	Path to folder for output files. Created if it does not exist. Defaults to `input_dir`.
`options`	An `av1r_options` list. Defaults to `av1r_options()`.
`ext`	Character vector of input extensions to process. Default: `c("mp4","avi","mkv","mov","flv","mpg","`
`skip_existing`	If `TRUE` (default), skip files where the output already exists.
`recursive`	If `TRUE`, scan subfolders up to `max_depth` levels deep. Default: `FALSE`.
`max_depth`	Maximum subfolder depth when `recursive = TRUE`. Default: 5.

Details

When a folder contains only single-page TIFF images (e.g. a microscopy image sequence), they are automatically combined into a single video named after the input folder.

When `recursive = TRUE`, subfolders (up to `max_depth` levels) are scanned. Each subfolder containing only TIFF images is combined into a single video named after the subfolder. Subfolders with video files are processed file-by-file. All output is written flat into `output_dir`, with subfolder names used as prefixes to avoid collisions.

Value

Invisibly returns a data.frame with columns `input`, `output`, `input_size` (bytes), `status` ("ok", "skipped", or "error"), and `message`.

Examples

```
## Not run:
# Requires FFmpeg installed
convert_folder("~/data/microscopy", file.path(tempdir(), "av1_output"))

# Recursive: each subfolder with TIFFs becomes a separate video
convert_folder("~/data/plates", "~/av1_output", recursive = TRUE)

## End(Not run)
```

`convert_to_av1` *Convert video to AV1*

Description

Converts biological microscopy video files (MP4/H.264, H.265, AVI/MJPEG) or TIFF stacks to AV1 format. Automatically selects the best available backend: Vulkan GPU (`VK_KHR_VIDEO_ENCODE_AV1`) when a compatible device is found, otherwise CPU via FFmpeg (`libsvtav1`).

Usage

```
convert_to_av1(input, output, options = av1r_options())
```

Arguments

`input`	Path to input file. Supported: .mp4, .avi, .mkv, .mov, .flv, .mpg, .mpeg, .webm, .tif/.tiff (multi-page), or printf pattern like `"frame%04d.tif"`.
`output`	Path to output file (.mp4 or .mkv).
`options`	An `av1r_options` list. Defaults to `av1r_options()`. Use `backend = "cpu"` or `backend = "vulkan"` to force a backend.

Value

Invisibly returns 0L on success. Stops with an error on failure.

Examples

```
# List available options
str(av1r_options())

## Not run:
# Requires FFmpeg installed
convert_to_av1("recording.mp4", file.path(tempdir(), "recording_av1.mp4"))

## End(Not run)
```

detect_backend — *Detect best available encoding backend*

Description

Priority order: "vaapi" (VAAPI AV1 GPU via ffmpeg, best speed/quality ratio) > "vulkan" (Vulkan AV1 GPU, fastest but CQP only) > "cpu" (SVT-AV1 via ffmpeg).

Usage

```
detect_backend(prefer = "auto")
```

Arguments

prefer	"auto" (default), "vaapi", "vulkan", or "cpu".

Value

Character string: "vaapi", "vulkan", or "cpu".

measure_ssim — *Measure SSIM quality score between two video files*

Description

Compares encoded video against the original using SSIM. Values close to 1.0 indicate high similarity.

Usage

```
measure_ssim(original, encoded, duration = NULL)
```

Arguments

original	Path to original video file.
encoded	Path to encoded video file.
duration	Seconds to compare. NULL = full video.

Value

Numeric SSIM score (0-1), or NA on failure.

read_tiff_stack *Read a TIFF stack and return basic metadata*

Description

Does not load pixel data into R – only inspects the file. Useful for checking frame count and dimensions before encoding.

Usage

```
read_tiff_stack(path)
```

Arguments

path	Path to a multi-page TIFF file or printf-pattern (e.g. "frame%04d.tif").

Value

A list with elements: path, n_frames, width, height, size_mb.

Examples

```
# Create a small temporary TIFF and inspect it
tmp <- tempfile(fileext = ".tif")
writeBin(raw(1024), tmp)
info <- read_tiff_stack(tmp)
str(info)
unlink(tmp)
```

vulkan_available *Check Vulkan AV1 availability*

Description

Check Vulkan AV1 availability

Usage

```
vulkan_available()
```

Value

TRUE if the package was compiled with Vulkan AV1 encode support.

`vulkan_devices` *List Vulkan-capable GPU devices*

Description

List Vulkan-capable GPU devices

Usage

```
vulkan_devices()
```

Value

Character vector of device names. Devices supporting `VK_KHR_VIDEO_ENCODE_AV1` are marked with `[AV1]`.

Index

AV1R *(AV1R-package)*, 2
AV1R-package, 2
av1r_options, 3
av1r_status, 4

convert_folder, 5
convert_to_av1, 6

detect_backend, 7

measure_ssim, 7

read_tiff_stack, 8

vulkan_available, 8
vulkan_devices, 9