Package ‘picohdr’

October 28, 2024

Type Package

Title Read, Write and Manipulate High Dynamic Range Images

Version 0.1.1

Maintainer Mike Cheng <mikefc@coolbutuseless.com>

Description High Dynamic Range (HDR) images support a large range in luminosity between the lightest and darkest regions of an image. To capture this range, data in HDR images is often stored as floating point numbers and in formats that capture more data and channels than standard image types. This package supports reading and writing two types of HDR images; PFM (Portable Float Map) and OpenEXR images. HDR images can be converted to lower dynamic ranges (for viewing) using tone-mapping. A number of tone-mapping algorithms are included which are based on Reinhard (2002) ``Photographic tone reproduction for digital images'' <doi:10.1145/566654.566575>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports ctypesio

Suggests knitr, rmarkdown, ggplot2, testthat (>= 3.0.0)

Config/testthat/edition 3

Copyright The included images in the 'inst/image' directory were created by the package author (Mike Cheng) and are licensed under CC0 1.0. See 'COPYRIGHTS' file for more details.

VignetteBuilder knitr

URL https://github.com/coolbutuseless/picohdr

BugReports https://github.com/coolbutuseless/picohdr/issues

NeedsCompilation yes

Author Mike Cheng [aut, cre, cph]

Repository CRAN

Date/Publication 2024-10-28 12:00:08 UTC

Contents

`adj_clamp` *Clamp values outside the specified range*

Description

Clamp values outside the specified range

Usage

```
adj_clamp(arr, lo = -Inf, hi = Inf)
```

Arguments

`arr`	array or matrix
`lo`	low value. Values lower than this will be replaced with this value. Default: -Inf
`hi`	Values higher than this will be replaced with this value. Default: Inf

Value

adjusted array

See Also

Other array adjustment functions: `adj_gamma()`, `adj_infinite()`, `adj_rescale()`, `adj_shift_negatives_global()`, `adj_shift_negatives_local()`

Examples

```
arr <- array(1:12, c(4, 3, 1))
arr
adj_clamp(arr, 10, 20)
```

---

adj_gamma *Adjust gamma*

Description

Adjust gamma

Usage

```
adj_gamma(arr, gamma = 1/2.2)
```

Arguments

`arr`	array or matrix
`gamma`	gamma correction factor. Default: 1/2.2

Value

adjusted array

See Also

Other array adjustment functions: `adj_clamp()`, `adj_infinite()`, `adj_rescale()`, `adj_shift_negatives_global()`, `adj_shift_negatives_local()`

Examples

```
arr <- array(1:12, c(4, 3, 1))
arr
adj_gamma(arr)
```

adj_infinite — *Replace infinite values with the minimum/maximum of the finite values*

Description

Replace infinite values with the minimum/maximum of the finite values

Usage

```
adj_infinite(arr)
```

Arguments

arr	array or matrix

Value

adjusted array

See Also

Other array adjustment functions: adj_clamp(), adj_gamma(), adj_rescale(), adj_shift_negatives_global(), adj_shift_negatives_local()

Examples

```
arr <- array(c(-Inf, Inf, 1:10), c(4, 3, 1))
arr
adj_infinite(arr)
```

adj_rescale — *Linearly rescale values to lie between the given limits*

Description

Infinite values will be clamped to the limits

Usage

```
adj_rescale(arr, lo, hi)
```

Arguments

arr	array or matrix
lo, hi	limits

Value

adjusted array

See Also

Other array adjustment functions: `adj_clamp()`, `adj_gamma()`, `adj_infinite()`, `adj_shift_negatives_global()`, `adj_shift_negatives_local()`

Examples

```
arr <- array(1:24, c(4, 3, 2))
arr
adj_rescale(arr, 0, 1)
```

`adj_shift_negatives_global`
Shift all values such that the minimum of the array is 0

Description

Shift all values such that the minimum of the array is 0

Usage

```
adj_shift_negatives_global(arr)
```

Arguments

`arr`	array or matrix

Value

adjusted array

See Also

Other array adjustment functions: `adj_clamp()`, `adj_gamma()`, `adj_infinite()`, `adj_rescale()`, `adj_shift_negatives_local()`

Examples

```
arr <- array(c(-5, 1:23), c(4, 3, 2))
arr
adj_shift_negatives_global(arr)
```

`adj_shift_negatives_local` *Shift all values in a plane such that the minimum in every plane is 0*

Description

Shift all values in a plane such that the minimum in every plane is 0

Usage

```
adj_shift_negatives_local(arr)
```

Arguments

`arr`	array or matrix

Value

adjusted array

See Also

Other array adjustment functions: `adj_clamp()`, `adj_gamma()`, `adj_infinite()`, `adj_rescale()`, `adj_shift_negatives_global()`

Examples

```
arr <- array(c(-5, 1:23), c(4, 3, 2))
arr
adj_shift_negatives_local(arr)
```

`array_to_df` *Convert array to a linear data.frame. Preserves array names if present.*

Description

This conversion is useful when preparing the data to summarise with `ggplot`.

Usage

```
array_to_df(arr)
```

Arguments

`arr`	array

Value

data.frame with 'x', 'y', 'z', 'channel' and 'value.' 'channel' will be the channel name if found, otherwise it is equivalent to 'z'.

Examples

```
arr <- array(1:24, dim = c(4, 3, 2))
array_to_df(arr)
```

`exr_attrs` *Helper function to create attributes for* `write_exr()`

Description

The EXR file specification requires particular types to define the metadata for the image. This function helps define these metadata attributes.

Usage

```
exr_attrs(
  channels = NULL,
  compression = NULL,
  dataWindow = NULL,
  displayWindow = NULL,
  lineOrder = NULL,
  pixelAspectRatio = NULL,
  screenWindowCenter = NULL,
  screenWindowWidth = NULL,
  ...
)
```

Arguments

`channels`	`[exr_type$chlist()]` data.frame of channel information with columns name [string], type ['half', 'float', 'uint'], pLinear [0, 1], xSampling [0, 1], ySampling [0, 1]
`compression`	`[exr_type$compression()]` 'NONE' or 'ZIP'
`dataWindow`	`[exr_type$box2i()]` xmin, ymin, xmax, ymax of data. Default: image size c(0, 0, w-1, h-1)
`displayWindow`	`[exr_type$box2i()]` xmin, ymin, xmax, ymax of display. Default: image size c(0, 0, w-1, h-1)
`lineOrder`	`[exr_type$lineOrder()]` Line ordering. One of 'increasing', 'decreasing' or 'random'. Default: 'increasing'
`pixelAspectRatio`	`[exr_type$float()]`. Default: 1.0

`screenWindowCenter`	`[exr_type$v2f()]`. Default: c(0.0, 0.0)
`screenWindowWidth`	`[exr_type$float()]`. Default: 1.0
`...`	Other named parameters. value must be of class `exr_type` e.g. `myLabel = exr_type$string("potpourri")`.

Details

In the majority of cases for basic image output, there is no need to specify anything with this function. `write_exr()` will create mandatory attributes required for image output.

Note that all values must be an object with class `exr_type`. To create these types, use `exr_type$<TYPE>(...)`.

Value

named list of attributes for writing EXR

Examples

```
exr_attrs(compression = exr_type$compression("ZIP"),
          name        = exr_type$string("Render 032"))
```

`exr_info` *Extract the metadata from an EXR file*

Description

This will extract attributes from any of EXR file.

Usage

```
exr_info(filename, verbosity = 0)
```

Arguments

`filename`	EXR filename or connection
`verbosity`	verbosity. Default: 0

Value

Named list of image attributes

Examples

```
filename <- system.file("image/rstats.exr", package = "picohdr")
exr_info(filename)
```

exr_type — *Functions for creating valid EXR type objects*

Description

This is a list of functions for creating EXR objects of a particular EXR type. Each function does checks for argument validity and calculates size information required for EXR output.

Usage

```
exr_type
```

Format

An object of class `list` of length 23.

Details

Refer to official OpenEXR documentation

Examples

```
# Create a v2f type
exr_type$v2f(c(12.1, 2.3))

# Create an attribute
exr_attrs(copyright = exr_type$string("mike"))
```

plot.array — *Plot method for matrices and arrays*

Description

Plot method for matrices and arrays

Usage

```
## S3 method for class 'array'
plot(x, interpolate = TRUE, ...)
```

Arguments

`x`	matrix or array
`interpolate`	Default: TRUE
`...`	other arguments passed to plot()

Value

None.

Examples

```
filename <- system.file("image/rstats.pfm.bz2", package = "picohdr")
image <- read_pfm(filename)
image <- adj_gamma(image)
plot(image)
```

`print.exr_type` *Print 'exr_type' objects*

Description

Print 'exr_type' objects

Usage

```
## S3 method for class 'exr_type'
print(x, ...)
```

Arguments

`x`	exr_type object
`...`	other arguments passed on to NextMethod

Value

None

Examples

```
bbox <- exr_type$box2i(0, 0, 1, 1)
print(bbox)
```

`read_exr` *Read an EXR image*

Description

Currently only single-part scanline images are supported (where the compression is one of NONE, ZIP or ZIPS).

Usage

```
read_exr(filename, verbosity = 0)
```

Arguments

`filename`	EXR filename or connection
`verbosity`	Level of debugging output. Default: 0 (no debugging output)

Value

Numeric array with names along the third dimension. Each plane in the array corresponds to a channel in the EXR.

Examples

```
filename <- system.file("image/rstats.exr", package = "picohdr")
images <- read_exr(filename)
dimnames(images)[[3]]

# Naively adjust one of the images for display
im <- adj_rescale(images[, , 'dzdy'], lo = 0, hi = 1)
plot(im)
```

`read_pfm` *Read PFM image*

Description

Read PFM image

Usage

```
read_pfm(filename)
```

Arguments

`filename`	PFM filename or connection object. If filename ends with ’xz’, ’bz2’ or ’gz’ suffix then it will be uncompressed automatically.

Value

If input PFM file is grayscale, a 2D numeric array is returned. If PFM file represents RGB color values, a 3D numeric array is returned.

See Also

Other PFM functions: `write_pfm()`

Examples

```
file <- system.file("image/rstats.pfm.bz2", package = "picohdr")
arr <- read_pfm(file)
arr[1:5, 1:5, ]

# Tone-map the image, gamma correct and plot
arr <- tm_reinhard_basic(arr)
arr <- adj_gamma(arr)
plot(arr)
```

`tm_reinhard` *Reinhard's global tone mapping*

Description

Tone mapping is a method for adapting an HDR image for display on a low dynamic range device. There are three included variants of Reinhard's global tone mapping operator.

Usage

```
tm_reinhard(arr)

tm_reinhard_basic(arr)

tm_reinhard_variant(arr)
```

Arguments

`arr`	array or matrix

Details

`tm_reinhard()` [RGB] Reinhard's operator with a correction for the maximum luminance

`tm_reinhard_basic()` [RGB images] Reinhard's operator applied equally to all colour channels

`tm_reinhard_variant()` [RGB or Gray images] A combination of the above two methods

These functions are based on Reinhard (2002) "Photographic tone reproduction for digital images"

Value

New array with adjusted color values

Examples

```
filename <- system.file("image", "rstats.pfm.bz2", package = "picohdr")
image <- read_pfm(filename)
image <- tm_reinhard_basic(image)
image <- adj_gamma(image)
plot(image)
```

---

write_exr *Write a numeric array as an EXR image*

Description

Write a numeric array as an EXR image

Usage

```
write_exr(
  arr,
  filename,
  pixel_type = c("half", "float", "uint"),
  channel_names = NULL,
  attrs = exr_attrs(),
  verbosity = 0
)
```

Arguments

`arr`	array representing image
`filename`	filename
`pixel_type`	one of 'half', 'float' or 'double'. Default: 'half'
`channel_names`	character vector. names of each plane in the array. If `NULL` then channel names are extracted from the array with `dimnames(arr)[[3]]`. If no names are set on the array, then channel names defaults to "Y", "RGB" and "RGBA" for 1, 3, and 4 plane arrays respectively. For all other array sizes, channel names allocated alphabetically from 'A' to 'Z'
`attrs`	EXR attributes for image. Use `exr_attrs()`
`verbosity`	verbosity. default: 0

Value

None

Examples

```
orig_file <- system.file("image", "rstats.pfm.bz2", package = "picohdr")
arr <- read_pfm(orig_file)
exr_file <- tempfile(fileext = ".exr")
write_exr(arr, exr_file)
```

`write_pfm` *Write a numeric array as PFM*

Description

Write a numeric array as PFM

Usage

```
write_pfm(arr, filename, endian = "little")
```

Arguments

`arr`	numeric matrix or array (with 3 planes)
`filename`	filename or connection object. If filename ends with ".xz", '.bz2' or '.gz', then it will be automatically compressed.
`endian`	One of 'little' or 'big'. Default: 'little'

Value

None.

See Also

Other PFM functions: `read_pfm()`

Examples

```
arr <- array(runif(10 * 30 * 3), dim = c(10, 30, 3))
write_pfm(arr, tempfile())
```

Index