Package 'nandb'

May 8, 2020

Type Package

Title Number and Brightness Image Analysis

Version 2.0.7

Maintainer Rory Nolan <rorynoolan@gmail.com>

Description Calculation of molecular number and brightness from fluorescence microscopy image series. The software was published in a 2016 paper <doi:10.1093/bioinformatics/btx434>. The seminal paper for the technique is Digman et al. 2008 <doi:10.1529/biophysj.107.114645>. A review of the technique was published in 2017 <doi:10.1016/j.ymeth.2017.12.001>.

License BSD_3_clause + file LICENSE

URL https://rorynolan.github.io/nandb, https://github.com/rorynolan/nandb

BugReports https://github.com/rorynolan/nandb/issues

Depends R (>= 3.1)

Imports assertthat, autothresholdr (>= 1.3.3), BBmisc, checkmate (>= 1.9.3), detrendr (>= 0.6.2), dplyr, filesstrings (>= 3.1.5), ggplot2, glue (>= 1.3.0), ijtiff (>= 2.0.2), magrittr (>= 1.5), purrr, Rcpp (>= 1.0.1), reshape2, rlang (>= 0.3.3), stringr (>= 1.2.0), utils, viridis, withr (>= 2.1.0)

Suggests abind, covr, gridExtra, knitr, matrixStats (>= 0.50.0), pacman, rmarkdown, spelling, testthat (>= 2.1.0), tidyr

LinkingTo Rcpp (>= 1.0.1)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData TRUE

RoxygenNote 7.1.0

NeedsCompilation yes

Author Rory Nolan [aut, cre, cph] (<https://orcid.org/0000-0002-5239-4043>),
Luis Alvarez [ctb, cph] (<https://orcid.org/0000-0003-1316-1906>),
Sergi Padilla-Parra [ctb, ths, cph]
(<https://orcid.org/0000-0002-8010-9481>)

Repository CRAN

Date/Publication 2020-05-08 13:10:06 UTC

R topics documented:

`brightness` *Calculate brightness from image series.*

Description

Given a time stack of images, `brightness()` performs a calculation of the brightness for each pixel.

Usage

```
brightness(
  img,
  def,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  s = 1,
  offset = 0,
  readout_noise = 0,
  parallel = FALSE
)
```

Arguments

`img`	A 4-dimensional array in the style of an ijtiff_img (indexed by `img[y,x,channel,frame]`) or a 3-dimensional array which is a single channel of an ijtiff_img (indexed by `img[y,x,frame]`).
`def`	A character. Which definition of brightness do you want to use, `"B"` or `"epsilon"`?
`thresh`	The threshold or thresholding method (see `autothresholdr::mean_stack_thresh()`) to use on the image prior to detrending and brightness calculations.
`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. If there are many channels, this may be specified as a vector, one element for each channel.
`quick`	If `FALSE` (the default), the swap finding routine is run several times to get a consensus for the best parameter. If `TRUE`, the swap finding routine is run only once.
`filt`	Do you want to smooth (`filt = 'mean'`) or median (`filt = 'median'`) filter the number image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 (with corners included, so each median is the median of 9 elements) and with the option `na_count = TRUE`. If you want to smooth/median filter the number image in a different way, first calculate the numbers without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result. If there are many channels, this may be specified as a vector, one element for each channel.
`s`	A positive number. The S-factor of microscope acquisition.
`offset`	Microscope acquisition parameters. See reference Dalal et al.
`readout_noise`	Microscope acquisition parameters. See reference Dalal et al.
`parallel`	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

Value

A matrix, the brightness image.

References

Digman MA, Dalal R, Horwitz AF, Gratton E. Mapping the Number of Molecules and Brightness in the Laser Scanning Microscope. Biophysical Journal. 2008;94(6):2320-2332. doi: 10.1529/biophysj.107.114645.

Dalal, RB, Digman, MA, Horwitz, AF, Vetri, V, Gratton, E (2008). Determination of particle number and brightness using a laser scanning confocal microscope operating in the analog mode. Microsc. Res. Tech., 71, 1:69-81. doi: 10.1002/jemt.20526.

Examples

```
img <- ijtiff::read_tif(system.file("extdata", "50.tif", package = "nandb"))
ijtiff::display(img[, , 1, 1])
b <- brightness(img, "e", thresh = "Huang")
b <- brightness(img, "B", thresh = "tri")
```

`brightness_folder` *Brightness calculations for every image in a folder.*

Description

Perform `brightness()` calculations on all tif images in a folder and save the resulting brightness images to disk.

Usage

```
brightness_folder(
  folder_path = ".",
  def,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  s = 1,
  offset = 0,
  readout_noise = 0,
  parallel = FALSE
)
```

Arguments

`folder_path`	The path (relative or absolute) to the folder you wish to process.
`def`	A character. Which definition of brightness do you want to use, `"B"` or `"epsilon"`?
`thresh`	The threshold or thresholding method (see `autothresholdr::mean_stack_thresh()`) to use on the image prior to detrending and brightness calculations.

`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. If there are many channels, this may be specified as a vector, one element for each channel.
`quick`	If `FALSE` (the default), the swap finding routine is run several times to get a consensus for the best parameter. If `TRUE`, the swap finding routine is run only once.
`filt`	Do you want to smooth (`filt = 'mean'`) or median (`filt = 'median'`) filter the number image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 (with corners included, so each median is the median of 9 elements) and with the option `na_count = TRUE`. If you want to smooth/median filter the number image in a different way, first calculate the numbers without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result. If there are many channels, this may be specified as a vector, one element for each channel.
`s`	A positive number. The S-factor of microscope acquisition.
`offset`	Microscope acquisition parameters. See reference Dalal et al.
`readout_noise`	Microscope acquisition parameters. See reference Dalal et al.
`parallel`	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

See Also

`number()`

Examples

```
## Not run:
setwd(tempdir())
img <- ijtiff::read_tif(system.file("extdata", "50.tif", package = "nandb"))
ijtiff::write_tif(img, "img1.tif")
ijtiff::write_tif(img, "img2.tif")
brightness_folder(def = "B", thresh = "Huang")

## End(Not run)
```

`brightness_timeseries` *Create a brightness time-series.*

Description

Given a stack of images `img`, use the first `frames_per_set` of them to create one brightness image, the next `frames_per_set` of them to create the next brightness image and so on to get a time-series of brightness images.

Usage

```
brightness_timeseries(
  img,
  def,
  frames_per_set,
  overlap = FALSE,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  s = 1,
  offset = 0,
  readout_noise = 0,
  parallel = FALSE
)
```

Arguments

`img`	A 4-dimensional array in the style of an ijtiff_img (indexed by `img[y,x,channel,frame]`) or a 3-dimensional array which is a single channel of an ijtiff_img (indexed by `img[y,x,frame]`).
`def`	A character. Which definition of brightness do you want to use, `"B"` or `"epsilon"`?
`frames_per_set`	The number of frames with which to calculate the successive brightnesses.
`overlap`	A boolean. If `TRUE`, the windows used to calculate number are overlapped, if `FALSE`, they are not. For example, for a 20-frame image series with 5 frames per set, if the windows are not overlapped, then the frame sets used are 1-5, 6-10, 11-15 and 16-20; whereas if they are overlapped, the frame sets are 1-5, 2-6, 3-7, 4-8 and so on up to 16-20.
`thresh`	The threshold or thresholding method (see `autothresholdr::mean_stack_thresh()`) to use on the image prior to detrending and brightness calculations.
`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. If there are many channels, this may be specified as a vector, one element for each channel.
`quick`	If `FALSE` (the default), the swap finding routine is run several times to get a consensus for the best parameter. If `TRUE`, the swap finding routine is run only once.
`filt`	Do you want to smooth (`filt = 'mean'`) or median (`filt = 'median'`) filter the number image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 (with corners included, so each median is the median of 9 elements) and with the option `na_count = TRUE`. If you want to smooth/median filter the number image in a different way, first calculate the numbers without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result. If there are many channels, this may be specified as a vector, one element for each channel.
`s`	A positive number. The S-factor of microscope acquisition.

`offset`	Microscope acquisition parameters. See reference Dalal et al.
`readout_noise`	Microscope acquisition parameters. See reference Dalal et al.
`parallel`	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

Details

This may discard some images, for example if 175 frames are in the input and `frames_per_set = 50`, then the last 25 are discarded. If detrending is selected, it is performed on the whole image stack before the sectioning is done for calculation of numbers.

Value

An object of class brightness_ts_img.

- If `img` is 3-dimensional (i.e. 1-channel), a 3-dimensional array `arr` is returned with `arr[y,x,t]` being pixel (x, y) of the tth brightness image in the brightness time series.
- If `img` is 4-dimensional (i.e. 2-channel), a 4-dimensional array `arr` is returned with `arr[y,x,c,t]` being pixel (x, y) of the cth channel of the tth brightness image in the brightness time series.

See Also

`brightness()`.

Examples

```
img <- ijtiff::read_tif(system.file("extdata", "50.tif", package = "nandb"))
bts <- brightness_timeseries(img, "e", frames_per_set = 20, thresh = "Huang")
```

`brightness_timeseries_folder`

Brightness time-series calculations for every image in a folder.

Description

Perform `brightness_timeseries()` calculations on all tif images in a folder and save the resulting number images to disk.

Usage

```
brightness_timeseries_folder(
  folder_path = ".",
  def,
  frames_per_set,
  overlap = FALSE,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  s = 1,
  offset = 0,
  readout_noise = 0,
  parallel = FALSE
)
```

Arguments

`folder_path`	The path (relative or absolute) to the folder you wish to process.
`def`	A character. Which definition of brightness do you want to use, `"B"` or `"epsilon"`?
`frames_per_set`	The number of frames with which to calculate the successive brightnesses.
`overlap`	A boolean. If `TRUE`, the windows used to calculate number are overlapped, if `FALSE`, they are not. For example, for a 20-frame image series with 5 frames per set, if the windows are not overlapped, then the frame sets used are 1-5, 6-10, 11-15 and 16-20; whereas if they are overlapped, the frame sets are 1-5, 2-6, 3-7, 4-8 and so on up to 16-20.
`thresh`	The threshold or thresholding method (see `autothresholdr::mean_stack_thresh()`) to use on the image prior to detrending and brightness calculations.
`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. If there are many channels, this may be specified as a vector, one element for each channel.
`quick`	If `FALSE` (the default), the swap finding routine is run several times to get a consensus for the best parameter. If `TRUE`, the swap finding routine is run only once.
`filt`	Do you want to smooth (`filt = 'mean'`) or median (`filt = 'median'`) filter the number image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 (with corners included, so each median is the median of 9 elements) and with the option `na_count = TRUE`. If you want to smooth/median filter the number image in a different way, first calculate the numbers without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result. If there are many channels, this may be specified as a vector, one element for each channel.
`s`	A positive number. The S-factor of microscope acquisition.
`offset`	Microscope acquisition parameters. See reference Dalal et al.

`readout_noise`	Microscope acquisition parameters. See reference Dalal et al.
`parallel`	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

See Also

`brightness_timeseries()`

Examples

```
## Not run:
setwd(tempdir())
img <- ijtiff::read_tif(system.file("extdata", "50.tif", package = "nandb"))
ijtiff::write_tif(img, "img1.tif")
ijtiff::write_tif(img, "img2.tif")
brightness_timeseries_folder(def = "e", thresh = "tri", frames_per_set = 20)

## End(Not run)
```

`cc-nb-img-classes` *Cross-correlated number and brightness image classes.*

Description

The `cc_number_img` and `cc_brightness_img` classes are designed to hold objects which are images calculated from the *cross-correlated number and brightness* technique.

Usage

```
cc_number_img(img, thresh, swaps, filt)

cc_brightness_img(img, thresh, swaps, filt)
```

Arguments

`img`	The calculated cross-correlated number or brightness image.
`thresh`	A positive integer, possibly an object of class autothresholdr::th. If the different channels of the image had different thresholds, this argument may be specified as a vector or list (of positive integers, possibly objects of class autothresholdr::th), one element for each channel.
`swaps`	A non-negative integer with an attribute `auto`. If the different channels of the image had different swaps, this argument may be specified as a list (of non-negative integers with attributes `auto`), one element for each channel. For undetrended images, set `swaps = NA`.
`filt`	A string, the filtering method used. Must be either `"mean"` or `"median"`, or `NA` for no filtering. If the different channels of the image had different filters, this may be specified as a character vector, one element for each channel.

Details

An object of class `cc_number_img` or `cc_brightness_img` is a 4-dimensional array of real numbers in the mould of an ijtiff_img (indexed as `img[y,x,channel,frame]`) with 3 attributes:

`thresh` A positive integer, possibly an object of class autothresholdr::th detailing which threshold and thresholding method was used in preprocessing (in the multi-channel case, one threshold per channel is given).

`swaps` A non-negative integer indicating the number of swaps used for Robin Hood detrending, with an attribute `auto` which is a logical indicating whether or not the parameter was chosen automatically (in the multi-channel case, one `swaps` per channel is given).

`filt` Was mean or median filtering used in postprocessing?

Value

An object of class `cc_number_img` or `cc_brightness_img`.

`cc-nb-ts-img-classes` *Cross-correlated number and brightness time series image classes.*

Description

The `cc_number_ts_img` and `cc_brightness_ts_img` classes are designed to hold objects which are images calculated from the *cross-correlated number and brightness* technique.

Usage

```
cc_number_ts_img(img, frames_per_set, overlapped, thresh, swaps, filt)

cc_brightness_ts_img(img, frames_per_set, overlapped, thresh, swaps, filt)
```

Arguments

`img`	The calculated cross-correlated number or brightness time series image series.
`frames_per_set`	The number of frames used in the calculation of each point in the cross-correlated number or brightness time series.
`overlapped`	A boolean. `TRUE` indicates that the windows used to calculate consecutive brightnesses over time were overlapped, `FALSE` indicates that they were not.
`thresh`	A positive integer, possibly an object of class autothresholdr::th. If the different channels of the image had different thresholds, this argument may be specified as a vector or list (of positive integers, possibly objects of class autothresholdr::th), one element for each channel.
`swaps`	A non-negative integer with an attribute `auto`. If the different channels of the image had different `swaps`, this argument may be specified as a list (of non-negative integers with attributes `auto`), one element for each channel. For undetrended images, set `swaps = NA`.
`filt`	A string, the filtering method used. Must be either `"mean"` or `"median"`, or `NA` for no filtering. If the different channels of the image had different filters, this may be specified as a character vector, one element for each channel.

Details

An object of class `cc_number_ts_img` or `cc_brightness_ts_img` is a 4-dimensional array of real numbers in the mould of an ijtiff_img with 3 attributes:

`thresh` A positive integer, possibly an object of class autothresholdr::th detailing which threshold and thresholding method was used in preprocessing (in the multi-channel case, one threshold per channel is given).

`swaps` A non-negative integer indicating the parameter used for Robin Hood detrending with an attribute `auto` which is a logical indicating whether or not the parameter was chosen automatically (in the multi-channel case, one `swaps` per channel is given).

`frames_per_set` A positive integer detailing how many frames were used in the calculation of each point in the number or brightness time series.

`overlapped` A boolean. `TRUE` indicates that the windows used to calculate consecutive brightnesses over time were overlapped, `FALSE` indicates that they were not.

Value

An object of class `cc_number_ts_img` or `cc_brightness_ts_img`.

See Also

`cc_number_timeseries()`, `cc_brightness_timeseries()`.

`cc_brightness` *Cross-correlated brightness.*

Description

Given a time stack of images and two channels, calculate the cross-correlated brightness of those two channels for each pixel.

Usage

```
cc_brightness(
  img,
  ch1 = 1,
  ch2 = 2,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  parallel = FALSE
)
```

Arguments

img	A 4-dimensional array of images indexed by img[y,x,channel,frame] (an object of class ijtiff::ijtiff_img). The image to perform the calculation on. To perform this on a file that has not yet been read in, set this argument to the path to that file (a string).
ch1	A natural number. The index of the first channel to use.
ch2	A natural number. The index of the second channel to use.
thresh	Do you want to apply an intensity threshold prior to calculating cross-correlated brightness (via autothresholdr::mean_stack_thresh())? If so, set your thresholding method here. If this is a single value, that same threshold will be applied to both channels. If this is a length-2 vector or list, then these two thresholds will be applied to channels 1 and 2 respectively. A value of NA for either channel gives no thresholding for that channel.
detrend	Detrend your data with detrendr::img_detrend_rh(). This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the detrendr package. To detrend one channel and not the other, specify this as a length 2 vector.
quick	FALSE repeats the detrending procedure (which has some inherent randomness) a few times to hone in on the best detrend. TRUE is quicker, performing the routine only once. FALSE is better.
filt	Do you want to smooth (filt = 'smooth') or median (filt = 'median') filter the cross-correlated brightness image using smooth_filter() or median_filter() respectively? If selected, these are invoked here with a filter radius of 1 and with the option na_count = TRUE. A value of NA for either channel gives no thresholding for that channel. If you want to smooth/median filter the cross-correlated brightness image in a different way, first calculate the cross-correlated brightnesses without filtering (filt = NULL) using this function and then perform your desired filtering routine on the result.
parallel	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use parallel = TRUE.

Value

A numeric matrix, the cross-correlated brightness image.

Examples

```
img <- ijtiff::read_tif(system.file("extdata", "two_ch.tif",
  package = "nandb"
))
ijtiff::display(detrendr::mean_pillars(img[, , 1, ]))
ijtiff::display(detrendr::mean_pillars(img[, , 2, ]))
b <- brightness(img, def = "e", thresh = "Huang", filt = "median")
ijtiff::display(b[, , 1, 1])
ijtiff::display(b[, , 2, 1])
cc_b <- cc_brightness(img, thresh = "Huang")
```

```
ijtiff::display(cc_b[, , 1, 1])
```

`cc_brightness_folder` *Cross-correlated brightness calculations for every image in a folder.*

Description

Perform `cc_brightness()` calculations on all TIFF images in a folder and save the resulting images to disk.

Usage

```
cc_brightness_folder(
  folder_path = ".",
  ch1 = 1,
  ch2 = 2,
  thresh = NULL,
  detrend = detrend,
  quick = quick,
  filt = NULL,
  parallel = FALSE
)
```

Arguments

`folder_path`	The path (relative or absolute) to the folder you wish to process.
`ch1`	A natural number. The index of the first channel to use.
`ch2`	A natural number. The index of the second channel to use.
`thresh`	Do you want to apply an intensity threshold prior to calculating cross-correlated brightness (via `autothresholdr::mean_stack_thresh()`)? If so, set your thresholding method here. If this is a single value, that same threshold will be applied to both channels. If this is a length-2 vector or list, then these two thresholds will be applied to channels 1 and 2 respectively. A value of `NA` for either channel gives no thresholding for that channel.
`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. To detrend one channel and not the other, specify this as a length 2 vector.
`quick`	`FALSE` repeats the detrending procedure (which has some inherent randomness) a few times to hone in on the best detrend. `TRUE` is quicker, performing the routine only once. `FALSE` is better.
`filt`	Do you want to smooth (`filt = 'smooth'`) or median (`filt = 'median'`) filter the cross-correlated brightness image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 and with

the option `na_count = TRUE`. A value of `NA` for either channel gives no thresholding for that channel. If you want to smooth/median filter the cross-correlated brightness image in a different way, first calculate the cross-correlated brightnesses without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result.

parallel	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

Examples

```
## Not run:
setwd(tempdir())
ijtiff::write_tif(img, "a.tif")
ijtiff::write_tif(img, "ab.tif")
cc_brightness_folder()
list.files()

## End(Not run)
```

---

cc_brightness_timeseries

Create a cross-correlated brightness time-series.

Description

Given a stack of images `img`, use the first `frames_per_set` of them to create one cross-correlated brightness image, the next `frames_per_set` of them to create the next and so on to get a time-series of cross-correlated brightness images.

Usage

```
cc_brightness_timeseries(
  img,
  frames_per_set,
  overlap = FALSE,
  ch1 = 1,
  ch2 = 2,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  parallel = FALSE
)
```

Arguments

`img`	A 4-dimensional array of images indexed by `img[y,x,channel,frame]` (an object of class ijtiff::ijtiff_img). The image to perform the calculation on. To perform this on a file that has not yet been read in, set this argument to the path to that file (a string).
`frames_per_set`	The number of frames with which to calculate the successive cross-correlated brightnesses. This may discard some images, for example if 175 frames are in the input and `frames_per_set = 50`, then the last 25 are discarded. If bleaching or/and thresholding are selected, they are performed on the whole image stack before the sectioning is done for calculation of cross-correlated brightnesses.
`overlap`	A boolean. If `TRUE`, the windows used to calculate brightness are overlapped, if `FALSE`, they are not. For example, for a 20-frame image series with 5 frames per set, if the windows are not overlapped, then the frame sets used are 1-5, 6-10, 11-15 and 16-20; whereas if they are overlapped, the frame sets are 1-5, 2-6, 3-7, 4-8 and so on up to 16-20.
`ch1`	A natural number. The index of the first channel to use.
`ch2`	A natural number. The index of the second channel to use.
`thresh`	Do you want to apply an intensity threshold prior to calculating cross-correlated brightness (via `autothresholdr::mean_stack_thresh()`)? If so, set your thresholding method here. If this is a single value, that same threshold will be applied to both channels. If this is a length-2 vector or list, then these two thresholds will be applied to channels 1 and 2 respectively. A value of `NA` for either channel gives no thresholding for that channel.
`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. To detrend one channel and not the other, specify this as a length 2 vector.
`quick`	`FALSE` repeats the detrending procedure (which has some inherent randomness) a few times to hone in on the best detrend. `TRUE` is quicker, performing the routine only once. `FALSE` is better.
`filt`	Do you want to smooth (`filt = 'smooth'`) or median (`filt = 'median'`) filter the cross-correlated brightness image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 and with the option `na_count = TRUE`. A value of `NA` for either channel gives no thresholding for that channel. If you want to smooth/median filter the cross-correlated brightness image in a different way, first calculate the cross-correlated brightnesses without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result.
`parallel`	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

Value

An array where the ith slice is the ith cross-correlated brightness image.

See Also

brightness().

Examples

```
img <- ijtiff::read_tif(system.file("extdata", "two_ch.tif",
  package = "nandb"
))
cc_bts <- cc_brightness_timeseries(img, 10,
  thresh = "Huang",
  filt = "median", parallel = 2
)
ijtiff::display(cc_bts[, , 1, 1])
```

---

cc_brightness_timeseries_folder

Cross-correlated brightness time-series calculations for every image in a folder.

Description

Perform cc_brightness_timeseries() calculations on all tif images in a folder and save the resulting images to disk.

Usage

```
cc_brightness_timeseries_folder(
  folder_path = ".",
  frames_per_set,
  overlap = FALSE,
  ch1 = 1,
  ch2 = 2,
  thresh = NULL,
  detrend = detrend,
  quick = quick,
  filt = NULL,
  parallel = FALSE
)
```

Arguments

folder_path	The path (relative or absolute) to the folder you wish to process.
frames_per_set	The number of frames with which to calculate the successive cross-correlated brightnesses.

	This may discard some images, for example if 175 frames are in the input and `frames_per_set` = `50`, then the last 25 are discarded. If bleaching or/and thresholding are selected, they are performed on the whole image stack before the sectioning is done for calculation of cross-correlated brightnesses.
`overlap`	A boolean. If `TRUE`, the windows used to calculate brightness are overlapped, if `FALSE`, they are not. For example, for a 20-frame image series with 5 frames per set, if the windows are not overlapped, then the frame sets used are 1-5, 6-10, 11-15 and 16-20; whereas if they are overlapped, the frame sets are 1-5, 2-6, 3-7, 4-8 and so on up to 16-20.
`ch1`	A natural number. The index of the first channel to use.
`ch2`	A natural number. The index of the second channel to use.
`thresh`	Do you want to apply an intensity threshold prior to calculating cross-correlated brightness (via `autothresholdr::mean_stack_thresh()`)? If so, set your thresholding method here. If this is a single value, that same threshold will be applied to both channels. If this is a length-2 vector or list, then these two thresholds will be applied to channels 1 and 2 respectively. A value of `NA` for either channel gives no thresholding for that channel.
`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. To detrend one channel and not the other, specify this as a length 2 vector.
`quick`	`FALSE` repeats the detrending procedure (which has some inherent randomness) a few times to hone in on the best detrend. `TRUE` is quicker, performing the routine only once. `FALSE` is better.
`filt`	Do you want to smooth (`filt = 'smooth'`) or median (`filt = 'median'`) filter the cross-correlated brightness image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 and with the option `na_count = TRUE`. A value of `NA` for either channel gives no thresholding for that channel. If you want to smooth/median filter the cross-correlated brightness image in a different way, first calculate the cross-correlated brightnesses without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result.
`parallel`	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

See Also

`cc_brightness_timeseries()`

Examples

```
## Not run:
setwd(tempdir())
ijtiff::write_tif(img, "a.tif")
ijtiff::write_tif(img, "ab.tif")
cc_brightness_timeseries_folder(frames_per_set = 25)
list.files()
```

```
## End(Not run)
```

`cc_number` *Cross-correlated number.*

Description

Given a time stack of images and two channels, calculate the cross-correlated number of those two channels for each pixel.

Usage

```
cc_number(
  img,
  ch1 = 1,
  ch2 = 2,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  parallel = FALSE
)
```

Arguments

`img`	A 4-dimensional array of images indexed by `img[y,x,channel,frame]` (an object of class ijtiff::ijtiff_img). The image to perform the calculation on. To perform this on a file that has not yet been read in, set this argument to the path to that file (a string).
`ch1`	A natural number. The index of the first channel to use.
`ch2`	A natural number. The index of the second channel to use.
`thresh`	Do you want to apply an intensity threshold prior to calculating cross-correlated number (via `autothresholdr::mean_stack_thresh()`)? If so, set your thresholding method here. If this is a single value, that same threshold will be applied to both channels. If this is a length-2 vector or list, then these two thresholds will be applied to channels 1 and 2 respectively. A value of `NA` for either channel gives no thresholding for that channel.
`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. To detrend one channel and not the other, specify this as a length 2 vector.
`quick`	`FALSE` repeats the detrending procedure (which has some inherent randomness) a few times to hone in on the best detrend. `TRUE` is quicker, performing the routine only once. `FALSE` is better.

`filt`	Do you want to smooth (`filt = 'smooth'`) or median (`filt = 'median'`) filter the cross-correlated number image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 and with the option `na_count = TRUE`. A value of `NA` for either channel gives no thresholding for that channel. If you want to smooth/median filter the cross-correlated number image in a different way, first calculate the cross-correlated numbers without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result.
`parallel`	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

Value

A numeric matrix, the cross-correlated number image.

Examples

```
img <- ijtiff::read_tif(system.file("extdata", "two_ch.tif",
  package = "nandb"
))
ijtiff::display(detrendr::mean_pillars(img[, , 1, ]))
ijtiff::display(detrendr::mean_pillars(img[, , 2, ]))
n <- number(img, def = "n", thresh = "Huang", filt = "median")
ijtiff::display(n[, , 1, 1])
ijtiff::display(n[, , 2, 1])
cc_n <- cc_number(img, thresh = "Huang")
ijtiff::display(cc_n[, , 1, 1])
```

`cc_number_folder` *Cross-correlated number calculations for every image in a folder.*

Description

Perform `cc_number()` calculations on all TIFF images in a folder and save the resulting images to disk.

Usage

```
cc_number_folder(
  folder_path = ".",
  ch1 = 1,
  ch2 = 2,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  parallel = FALSE
)
```

Arguments

folder_path	The path (relative or absolute) to the folder you wish to process.
ch1	A natural number. The index of the first channel to use.
ch2	A natural number. The index of the second channel to use.
thresh	Do you want to apply an intensity threshold prior to calculating cross-correlated number (via `autothresholdr::mean_stack_thresh()`)? If so, set your thresholding method here. If this is a single value, that same threshold will be applied to both channels. If this is a length-2 vector or list, then these two thresholds will be applied to channels 1 and 2 respectively. A value of `NA` for either channel gives no thresholding for that channel.
detrend	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. To detrend one channel and not the other, specify this as a length 2 vector.
quick	`FALSE` repeats the detrending procedure (which has some inherent randomness) a few times to hone in on the best detrend. `TRUE` is quicker, performing the routine only once. `FALSE` is better.
filt	Do you want to smooth (`filt = 'smooth'`) or median (`filt = 'median'`) filter the cross-correlated number image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 and with the option `na_count = TRUE`. A value of `NA` for either channel gives no thresholding for that channel. If you want to smooth/median filter the cross-correlated number image in a different way, first calculate the cross-correlated numbers without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result.
parallel	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

Examples

```
## Not run:
setwd(tempdir())
ijtiff::write_tif(img, "a.tif")
ijtiff::write_tif(img, "ab.tif")
cc_number_folder()
list.files()

## End(Not run)
```

`cc_number_timeseries` *Create a cross-correlated number time-series.*

Description

Given a stack of images `img`, use the first `frames_per_set` of them to create one cross-correlated number image, the next `frames_per_set` of them to create the next and so on to get a time-series of cross-correlated number images.

Usage

```
cc_number_timeseries(
  img,
  frames_per_set,
  overlap = FALSE,
  ch1 = 1,
  ch2 = 2,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  parallel = FALSE
)
```

Arguments

`img`	A 4-dimensional array of images indexed by `img[y,x,channel,frame]` (an object of class ijtiff::ijtiff_img). The image to perform the calculation on. To perform this on a file that has not yet been read in, set this argument to the path to that file (a string).
`frames_per_set`	The number of frames with which to calculate the successive cross-correlated numbers. This may discard some images, for example if 175 frames are in the input and `frames_per_set = 50`, then the last 25 are discarded. If bleaching or/and thresholding are selected, they are performed on the whole image stack before the sectioning is done for calculation of cross-correlated numbers.
`overlap`	A boolean. If `TRUE`, the windows used to calculate brightness are overlapped, if `FALSE`, they are not. For example, for a 20-frame image series with 5 frames per set, if the windows are not overlapped, then the frame sets used are 1-5, 6-10, 11-15 and 16-20; whereas if they are overlapped, the frame sets are 1-5, 2-6, 3-7, 4-8 and so on up to 16-20.
`ch1`	A natural number. The index of the first channel to use.
`ch2`	A natural number. The index of the second channel to use.
`thresh`	Do you want to apply an intensity threshold prior to calculating cross-correlated number (via `autothresholdr::mean_stack_thresh()`)? If so, set your thresholding method here. If this is a single value, that same threshold will be applied to both channels. If this is a length-2 vector or list, then these two thresholds will be applied to channels 1 and 2 respectively. A value of `NA` for either channel gives no thresholding for that channel.
`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. To detrend one channel and not the other, specify this as a length 2 vector.
`quick`	`FALSE` repeats the detrending procedure (which has some inherent randomness) a few times to hone in on the best detrend. `TRUE` is quicker, performing the routine only once. `FALSE` is better.

`filt`	Do you want to smooth (`filt = 'smooth'`) or median (`filt = 'median'`) filter the cross-correlated number image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 and with the option `na_count = TRUE`. A value of `NA` for either channel gives no thresholding for that channel. If you want to smooth/median filter the cross-correlated number image in a different way, first calculate the cross-correlated numbers without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result.
`parallel`	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

Value

An array where the ith slice is the ith cross-correlated number image.

See Also

`number()`.

Examples

```
img <- ijtiff::read_tif(system.file("extdata", "two_ch.tif",
  package = "nandb"
))
cc_nts <- cc_number_timeseries(img, 10,
  thresh = "Huang",
  filt = "median", parallel = 2
)
ijtiff::display(cc_nts[, , 1, 1])
```

`cc_number_timeseries_folder`

Cross-correlated number time-series calculations for every image in a folder.

Description

Perform `cc_number_timeseries()` calculations on all tif images in a folder and save the resulting images to disk.

Usage

```
cc_number_timeseries_folder(
  folder_path = ".",
  frames_per_set,
  overlap = FALSE,
  ch1 = 1,
  ch2 = 2,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  parallel = FALSE
)
```

Arguments

`folder_path`	The path (relative or absolute) to the folder you wish to process.
`frames_per_set`	The number of frames with which to calculate the successive cross-correlated numbers. This may discard some images, for example if 175 frames are in the input and `frames_per_set = 50`, then the last 25 are discarded. If bleaching or/and thresholding are selected, they are performed on the whole image stack before the sectioning is done for calculation of cross-correlated numbers.
`overlap`	A boolean. If `TRUE`, the windows used to calculate brightness are overlapped, if `FALSE`, they are not. For example, for a 20-frame image series with 5 frames per set, if the windows are not overlapped, then the frame sets used are 1-5, 6-10, 11-15 and 16-20; whereas if they are overlapped, the frame sets are 1-5, 2-6, 3-7, 4-8 and so on up to 16-20.
`ch1`	A natural number. The index of the first channel to use.
`ch2`	A natural number. The index of the second channel to use.
`thresh`	Do you want to apply an intensity threshold prior to calculating cross-correlated number (via `autothresholdr::mean_stack_thresh()`)? If so, set your thresholding method here. If this is a single value, that same threshold will be applied to both channels. If this is a length-2 vector or list, then these two thresholds will be applied to channels 1 and 2 respectively. A value of `NA` for either channel gives no thresholding for that channel.
`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. To detrend one channel and not the other, specify this as a length 2 vector.
`quick`	`FALSE` repeats the detrending procedure (which has some inherent randomness) a few times to hone in on the best detrend. `TRUE` is quicker, performing the routine only once. `FALSE` is better.
`filt`	Do you want to smooth (`filt = 'smooth'`) or median (`filt = 'median'`) filter the cross-correlated number image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 and with the option `na_count = TRUE`. A value of `NA` for either channel gives no thresholding for that channel. If you want to smooth/median filter the cross-correlated number image in a different way, first calculate the cross-correlated numbers without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result.

parallel	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use parallel = TRUE.

See Also

cc_number_timeseries()

Examples

```
## Not run:
setwd(tempdir())
ijtiff::write_tif(img, "a.tif")
ijtiff::write_tif(img, "ab.tif")
cc_number_timeseries_folder(frames_per_set = 25)
list.files()

## End(Not run)
```

cross_var *Calculate the* cross-variance *of two vectors.*

Description

The cross-variance function is defined in the reference.

Usage

```
cross_var(x, y)
```

Arguments

x	A numeric vector.
y	A numeric vector with the same length as x.

Value

A number

References

Digman, MA, Wiseman, PW, Choi, C, Horwitz, AR, Gratton, E (2009). Stoichiometry of molecular complexes at adhesions in living cells. Proc. Natl. Acad. Sci. U.S.A., 106, 7:2170-5.

Examples

```
cross_var(0:3, 2:5)
```

`cross_var_pillars` *Calculate the* cross-variance *of corresponding pillars of 3d arrays.*

Description

The cross-variance function is defined in the reference.

Usage

```
cross_var_pillars(x, y)
```

Arguments

`x`	A 3-dimensional array.
`y`	A 3-dimensional array with the same dimensions as `x`.

Details

Pillar i, j of the 3-dimensional array `arr` is `arr[i,j,]`.

Value

A matrix.

Examples

```
x <- array(1:27, dim = rep(3, 3))
y <- array(0:26, dim = rep(3, 3))
cross_var_pillars(x, y)
```

`matrix_raster_plot` *Make a raster plot of a matrix.*

Description

Given a matrix `mat`, make a raster plot of the matrix whereby in the plot, the pixel at $x =$`i`, $y =$`j` has colour based on the value of `mat[i,j]` and the x axis points right and the y axis points down (see 'Details').

Usage

```
matrix_raster_plot(
  mat,
  scale_name = "scale",
  limits = NULL,
  ranges = NULL,
  range_names = NULL,
  colours = NULL,
  na_colour = "black",
  clip = FALSE,
  clip_low = FALSE,
  clip_high = FALSE,
  log_trans = FALSE,
  breaks = NULL,
  include_breaks = NULL
)
```

Arguments

`mat`	The matrix you wish to plot.
`scale_name`	A string. The title of the color scale on the right of the plot.
`limits`	This gives the user the option to set all values outside a certain range to their nearest value within that range (if `clip` = `TRUE`) or to `NA` (if `clip` = `FALSE`. For example, to set all values outside the range [1.5, 2.6) to `NA`, use limits = c(1.5, 2.6), clip = FALSE. The colour range will cover all values within these specified limits.
`ranges`	A numeric vector. If you want specific ranges of values to have the same color, specify these ranges via an increasing numeric vector. For example, if you want the ranges 0.5-1.2 and 1.2-3, use `ranges` = `c(0.5,1.2,3)`. If `ranges` is specified as a number (a numeric vector of length 1) `n`, this is equivalent to setting ranges to be `n` equal-length intervals within the range of the matrix, i.e. it is equivalent to setting ‘ranges = seq(min(mat), max(mat), length.out = n • 1). At most one of rangesandlimitsshould be set. If ranges is set, the behaviour for values which are not in any of the ranges are set by theclip arguments as in thelimits‘ argument.
`range_names`	A character vector. If your colour scale is discrete, here you can set the names which will label each range in the legend.
`colours`	If you have set `ranges`, here you may specify which colors you wish to colour each range. It must have the same length as the number of intervals you have specified in `ranges`. If you have not specified `ranges`, here you may specify the colours (to be passed to `ggplot2::scale_fill_gradientn()`) to create the continuous colour band. It is specified as a character vector, with the colors specified either as the values in `colors()` or as in the value of the `rgb()` function. Note that this allows the use of `grDevices::rainbow()` and friends. The default uses `viridis::viridis()`.
`na_colour`	Which colour should the `NA` pixels be? Default is black.

`clip`	If either `limits` or `ranges` are set (one should never set both), there may be values that fall outside the specified limits/ranges. If `clip = TRUE`, values outside these limits/ranges are set to their nearest values within them, but if `clip = FALSE`, these values are set to NA. Note that setting `clip = TRUE` is equivalent to setting both `clip_low` and `clip_high` to `TRUE`.
`clip_low`	Setting this to `TRUE` (and leaving `clip = FALSE`, `clip_high = FALSE`) will set all values falling below the specified limits/ranges to their nearest value within them, but all values falling above those limits/ranges will be set to `NA`.
`clip_high`	Setting this to `TRUE` (and leaving `clip = FALSE`, `clip_low = FALSE`) will set all values falling above the specified limits/ranges to their nearest value within them, but all values falling below those limits/ranges will be set to `NA`.
`log_trans`	Do you want to log-transform the colour scaling?
`breaks`	Where do you want tick marks to appear on the legend colour scale?
`include_breaks`	If you don't want to specify all the breaks, but you want some specific ones to be included on the legend colour scale, specify those specific ones here.

Value

In the graphics console, a raster plot (via `ggplot2::geom_raster()`) will appear with the matrix values represented as pixel colours, with a named scale bar.

Examples

```
img <- ijtiff::read_tif(system.file("extdata", "50.tif", package = "nandb"))
ijtiff::display(img[, , 1, 1])
matrix_raster_plot(img[, , 1, 1])
b <- brightness(img, def = "B", detrend = FALSE, thresh = "Huang")
matrix_raster_plot(b, scale_name = "brightness")
matrix_raster_plot(b, scale_name = "brightness", log_trans = TRUE)
matrix_raster_plot(b,
  scale_name = "brightness", log_trans = TRUE,
  include_breaks = 1.35
)
matrix_raster_plot(b,
  scale_name = "brightness", log_trans = TRUE,
  breaks = 1:3
)
matrix_raster_plot(b,
  scale_name = "brightness",
  ranges = seq(0.5, 3, length.out = 6),
  range_names = paste0(1:5, "mer")
)
matrix_raster_plot(b,
  scale_name = "brightness",
  ranges = seq(0.5, 3, length.out = 6),
  range_names = paste0(1:5, "mer"), log_trans = TRUE
)
matrix_raster_plot(b,
  scale_name = "brightness",
  include_breaks = 1.25, range_names = NULL,
  log_trans = FALSE
)
matrix_raster_plot(b,
  scale_name = "brightness",
  include_breaks = 1.25, log_trans = TRUE
)
matrix_raster_plot(b,
  scale_name = "brightness",
  limits = c(1, 1.25), clip = TRUE
)
matrix_raster_plot(b,
  scale_name = "brightness",
  include_breaks = 1.25
)
```

median_filter *Smooth and median filters with options for handling NAs.*

Description

These are alternatives to `EBImage::filter2()` and `EBImage::medianFilter()` for smooth and median filtering respectively. These functions have many options for dealing with `NA` values which `EBImage`'s functions lack.

Usage

```
median_filter(mat, size = 1L, na_rm = FALSE, na_count = FALSE)

smooth_filter(mat, size = 1L, na_rm = FALSE, na_count = FALSE)
```

Arguments

`mat`	A matrix (representing an image).
`size`	An integer; the median filter radius.
`na_rm`	Should `NA`s be ignored?
`na_count`	If this is TRUE, in each median calculation, if the majority of arguments are `NA`s, `NA` is returned but if the `NA`s are in the minority, they are ignored as in `median(x,na.rm = TRUE)`.

Details

The behavior at image boundaries is such as the source image has been padded with pixels whose values equal the nearest border pixel value.

Value

A matrix (the median filtered image).

Examples

```
m <- matrix(1:9, nrow = 3)
m[2:3, 2:3] <- NA
print(m)
median_filter(m)
median_filter(m, na_rm = TRUE)
median_filter(m, na_count = TRUE)

smooth_filter(m)
smooth_filter(m, na_rm = TRUE)
smooth_filter(m, na_count = TRUE)
```

`nandb` *nandb: Number and brightness in R.*

Description

The nandb package gives functions for calculation of molecular number and brightness from images, as detailed in Digman et al. 2008. It comes with an implementation of the novel 'automatic detrending' technique.

References

Digman MA, Dalal R, Horwitz AF, Gratton E. Mapping the Number of Molecules and Brightness in the Laser Scanning Microscope. Biophysical Journal. 2008;94(6):2320-2332. doi: 10.1529/biophysj.107.114645.

`nb-img-classes` *Number and brightness image classes.*

Description

The `number_img` and `brightness_img` classes are designed to hold objects which are images calculated from the *number and brightness* technique.

Usage

```
number_img(img, def, thresh, swaps, filt)

brightness_img(img, def, thresh, swaps, filt)
```

Arguments

`img`	The calculated number or brightness image.
`def`	The number or brightness definition used.
`thresh`	A positive integer, possibly an object of class autothresholdr::th. If the different channels of the image had different thresholds, this argument may be specified as a vector or list (of positive integers, possibly objects of class autothresholdr::th), one element for each channel.
`swaps`	A non-negative integer with an attribute `auto`. If the different channels of the image had different swaps, this argument may be specified as a list (of non-negative integers with attributes `auto`), one element for each channel. For undetrended images, set `swaps = NA`.
`filt`	A string, the filtering method used. Must be either `"mean"` or `"median"`, or `NA` for no filtering. If the different channels of the image had different filters, this may be specified as a character vector, one element for each channel.

Details

An object of class `number_img` or `brightness_img` is a 4-dimensional array of real numbers in the mould of an ijtiff_img (indexed as `img[y,x,channel,frame]`) with 4 attributes:

`def` Are we using the `"N"` or `"n"` definition of number, or the `"B"` or `"epsilon"` definition of brightness?

`thresh` A positive integer, possibly an object of class autothresholdr::th detailing which threshold and thresholding method was used in preprocessing (in the multi-channel case, one threshold per channel is given).

`swaps` A non-negative integer indicating the number of swaps Robin Hood detrending, with an attribute `auto` which is a logical indicating whether or not the parameter was chosen automatically (in the multi-channel case, one threshold per channel is given).

`filt` Was mean or median filtering used in postprocessing?

Value

An object of class `number_img` or `brightness_img`.

`nb-ts-img-classes` *Number and brightness time series image classes.*

Description

The `number_ts_img` and `brightness_ts_img` classes are designed to hold objects which are images calculated from the *number and brightness* technique.

Usage

```
number_ts_img(img, def, frames_per_set, overlapped, thresh, swaps, filt)

brightness_ts_img(img, def, frames_per_set, overlapped, thresh, swaps, filt)
```

Arguments

`img`	The calculated number or brightness time series image series.
`def`	The number or brightness definition used.
`frames_per_set`	The number of frames used in the calculation of each point in the number or brightness time series.
`overlapped`	A boolean. `TRUE` indicates that the windows used to calculate consecutive brightnesses over time were overlapped, `FALSE` indicates that they were not.
`thresh`	A positive integer, possibly an object of class autothresholdr::th. If the different channels of the image had different thresholds, this argument may be specified as a vector or list (of positive integers, possibly objects of class autothresholdr::th), one element for each channel.
`swaps`	A non-negative integer with an attribute `auto`. If the different channels of the image had different `swaps`, this argument may be specified as a list (of non-negative integers with attributes `auto`), one element for each channel. For undetrended images, set `swaps = NA`.
`filt`	A string, the filtering method used. Must be either `"mean"` or `"median"`, or `NA` for no filtering. If the different channels of the image had different filters, this may be specified as a character vector, one element for each channel.

Details

An object of class `number_ts_img` or `brightness_ts_img` is a 3- or 4-dimensional array of real numbers with 4 attributes:

`def` Are we using the `"N"` or `"n"` definition of number, or the `"B"` or `"epsilon"` definition of brightness?

`thresh` A positive integer, possibly an object of class autothresholdr::th detailing which threshold and thresholding method was used in preprocessing (in the multi-channel case, one threshold per channel is given).

`swaps` A non-negative integer indicating the number of swaps used for Robin Hood detrending, with an attribute `auto` which is a logical indicating whether or not the parameter was chosen automatically (in the multi-channel case, one `swaps` per channel is given).

`frames_per_set` A positive integer detailing how many frames were used in the calculation of each point in the number or brightness time series.

`overlapped` A boolean. `TRUE` indicates that the windows used to calculate consecutive brightnesses over time were overlapped, `FALSE` indicates that they were not.

Value

An object of class `number_ts_img` or `brightness_ts_img`.

See Also

`number_timeseries()`, `brightness_timeseries()`.

`number` *Calculate number from image series.*

Description

Given a time stack of images, `number()` performs a calculation of the number for each pixel.

Usage

```
number(
  img,
  def,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  s = 1,
  offset = 0,
  readout_noise = 0,
  gamma = 1,
  parallel = FALSE
)
```

Arguments

`img`	A 4-dimensional array of images indexed by `img[y,x,channel,frame]` (an object of class ijtiff::ijtiff_img). The image to perform the calculation on. To perform this on a file that has not yet been read in, set this argument to the path to that file (a string).
`def`	A character. Which definition of number do you want to use, `"n"` or `"N"`?
`thresh`	The threshold or thresholding method (see `autothresholdr::mean_stack_thresh()`) to use on the image prior to detrending and number calculations. If there are many channels, this may be specified as a vector or list, one element for each channel.
`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. If there are many channels, this may be specified as a vector, one element for each channel.
`quick`	`FALSE` repeats the detrending procedure (which has some inherent randomness) a few times to hone in on the best detrend. `TRUE` is quicker, performing the routine only once. `FALSE` is better.
`filt`	Do you want to smooth (`filt = 'mean'`) or median (`filt = 'median'`) filter the number image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 (with corners included, so each median is the median of 9 elements) and with the option `na_count =`

	`TRUE`. If you want to smooth/median filter the number image in a different way, first calculate the numbers without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result. If there are many channels, this may be specified as a vector, one element for each channel.
`s`	A positive number. The S-factor of microscope acquisition.
`offset, readout_noise`	Microscope acquisition parameters. See reference Dalal et al.
`gamma`	Factor for correction of number n due to the illumination profile. The default (`gamma = 1`) has no effect. Changing gamma will have the effect of dividing the result by `gamma`, so the result with `gamma = 0.5` is two times the result with `gamma = 1`. For a Gaussian illumination profile, use `gamma = 0.3536`; for a Gaussian-Lorentzian illumination profile, use `gamma = 0.0760`.
`parallel`	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

Value

A matrix, the number image.

References

Digman MA, Dalal R, Horwitz AF, Gratton E. Mapping the Number of Molecules and Brightness in the Laser Scanning Microscope. Biophysical Journal. 2008;94(6):2320-2332. doi: 10.1529/biophysj.107.114645.

Dalal, RB, Digman, MA, Horwitz, AF, Vetri, V, Gratton, E (2008). Determination of particle number and brightness using a laser scanning confocal microscope operating in the analog mode. Microsc. Res. Tech., 71, 1:69-81. doi: 10.1002/jemt.20526.

Hur K-H, Macdonald PJ, Berk S, Angert CI, Chen Y, Mueller JD (2014) Quantitative Measurement of Brightness from Living Cells in the Presence of Photodepletion. PLoS ONE 9(5): e97440. doi: 10.1371/journal.pone.0097440.

Examples

```
img <- ijtiff::read_tif(system.file("extdata", "50.tif", package = "nandb"))
ijtiff::display(img[, , 1, 1])
num <- number(img, "N", thresh = "Huang")
num <- number(img, "n", thresh = "tri")
```

`number_folder` *Number calculations for every image in a folder.*

Description

Perform `number()` calculations on all tif images in a folder and save the resulting number images to disk.

Usage

```
number_folder(
  folder_path = ".",
  def,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  s = 1,
  offset = 0,
  readout_noise = 0,
  gamma = 1,
  parallel = FALSE
)
```

Arguments

`folder_path`	The path (relative or absolute) to the folder you wish to process.
`def`	A character. Which definition of number do you want to use, `"n"` or `"N"`?
`thresh`	The threshold or thresholding method (see `autothresholdr::mean_stack_thresh()`) to use on the image prior to detrending and number calculations. If there are many channels, this may be specified as a vector or list, one element for each channel.
`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. If there are many channels, this may be specified as a vector, one element for each channel.
`quick`	`FALSE` repeats the detrending procedure (which has some inherent randomness) a few times to hone in on the best detrend. `TRUE` is quicker, performing the routine only once. `FALSE` is better.
`filt`	Do you want to smooth (`filt = 'mean'`) or median (`filt = 'median'`) filter the number image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 (with corners included, so each median is the median of 9 elements) and with the option `na_count = TRUE`. If you want to smooth/median filter the number image in a different way, first calculate the numbers without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result. If there are many channels, this may be specified as a vector, one element for each channel.
`s`	A positive number. The S-factor of microscope acquisition.
`offset`	Microscope acquisition parameters. See reference Dalal et al.
`readout_noise`	Microscope acquisition parameters. See reference Dalal et al.
`gamma`	Factor for correction of number n due to the illumination profile. The default (`gamma = 1`) has no effect. Changing gamma will have the effect of dividing the result by `gamma`, so the result with `gamma = 0.5` is two times the result with `gamma = 1`. For a Gaussian illumination profile, use `gamma = 0.3536`; for a Gaussian-Lorentzian illumination profile, use `gamma = 0.0760`.

`parallel`	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

Note

Extreme number values (of magnitude greater than 3.40282e+38) will be written to the TIFF file as `NA`, since TIFF files cannot handle such huge numbers.

See Also

`number()`

Examples

```
## Not run:
setwd(tempdir())
img <- ijtiff::read_tif(system.file("extdata", "50.tif", package = "nandb"))
ijtiff::write_tif(img, "img2.tif")
number_folder(def = "n", thresh = "Huang", parallel = 2)

## End(Not run)
```

`number_timeseries` *Create a number time-series.*

Description

Given a stack of images `img`, use the first `frames_per_set` of them to create one number image, the next `frames_per_set` of them to create the next number image and so on to get a time-series of number images.

Usage

```
number_timeseries(
  img,
  def,
  frames_per_set,
  overlap = FALSE,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  s = 1,
  offset = 0,
  readout_noise = 0,
  gamma = 1,
  parallel = FALSE
)
```

Arguments

`img`	A 4-dimensional array of images indexed by `img[y,x,channel,frame]` (an object of class ijtiff::ijtiff_img). The image to perform the calculation on. To perform this on a file that has not yet been read in, set this argument to the path to that file (a string).
`def`	A character. Which definition of number do you want to use, `"n"` or `"N"`?
`frames_per_set`	The number of frames with which to calculate the successive numbers.
`overlap`	A boolean. If `TRUE`, the windows used to calculate brightness are overlapped, if `FALSE`, they are not. For example, for a 20-frame image series with 5 frames per set, if the windows are not overlapped, then the frame sets used are 1-5, 6-10, 11-15 and 16-20; whereas if they are overlapped, the frame sets are 1-5, 2-6, 3-7, 4-8 and so on up to 16-20.
`thresh`	The threshold or thresholding method (see `autothresholdr::mean_stack_thresh()`) to use on the image prior to detrending and number calculations. If there are many channels, this may be specified as a vector or list, one element for each channel.
`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. If there are many channels, this may be specified as a vector, one element for each channel.
`quick`	`FALSE` repeats the detrending procedure (which has some inherent randomness) a few times to hone in on the best detrend. `TRUE` is quicker, performing the routine only once. `FALSE` is better.
`filt`	Do you want to smooth (`filt = 'mean'`) or median (`filt = 'median'`) filter the number image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 (with corners included, so each median is the median of 9 elements) and with the option `na_count = TRUE`. If you want to smooth/median filter the number image in a different way, first calculate the numbers without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result. If there are many channels, this may be specified as a vector, one element for each channel.
`s`	A positive number. The S-factor of microscope acquisition.
`offset`	Microscope acquisition parameters. See reference Dalal et al.
`readout_noise`	Microscope acquisition parameters. See reference Dalal et al.
`gamma`	Factor for correction of number n due to the illumination profile. The default (`gamma = 1`) has no effect. Changing gamma will have the effect of dividing the result by `gamma`, so the result with `gamma = 0.5` is two times the result with `gamma = 1`. For a Gaussian illumination profile, use `gamma = 0.3536`; for a Gaussian-Lorentzian illumination profile, use `gamma = 0.0760`.
`parallel`	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

Details

This may discard some images, for example if 175 frames are in the input and `frames_per_set = 50`, then the last 25 are discarded. If detrending is selected, it is performed on the whole image stack before the sectioning is done for calculation of numbers.

Value

An object of class number_ts_img.

See Also

`number()`.

Examples

```
img <- ijtiff::read_tif(system.file("extdata", "50.tif", package = "nandb"))
nts <- number_timeseries(img, "n", frames_per_set = 20, thresh = "Huang")
```

`number_timeseries_folder`
Number time-series calculations for every image in a folder.

Description

Perform `number_timeseries()` calculations on all tif images in a folder and save the resulting number images to disk.

Usage

```
number_timeseries_folder(
  folder_path = ".",
  def,
  frames_per_set,
  overlap = FALSE,
  thresh = NULL,
  detrend = FALSE,
  quick = FALSE,
  filt = NULL,
  s = 1,
  offset = 0,
  readout_noise = 0,
  gamma = 1,
  parallel = FALSE
)
```

Arguments

`folder_path`	The path (relative or absolute) to the folder you wish to process.
`def`	A character. Which definition of number do you want to use, `"n"` or `"N"`?
`frames_per_set`	The number of frames with which to calculate the successive numbers.
`overlap`	A boolean. If `TRUE`, the windows used to calculate brightness are overlapped, if `FALSE`, they are not. For example, for a 20-frame image series with 5 frames per set, if the windows are not overlapped, then the frame sets used are 1-5, 6-10, 11-15 and 16-20; whereas if they are overlapped, the frame sets are 1-5, 2-6, 3-7, 4-8 and so on up to 16-20.
`thresh`	The threshold or thresholding method (see `autothresholdr::mean_stack_thresh()`) to use on the image prior to detrending and number calculations. If there are many channels, this may be specified as a vector or list, one element for each channel.
`detrend`	Detrend your data with `detrendr::img_detrend_rh()`. This is the best known detrending method for brightness analysis. For more fine-grained control over your detrending, use the `detrendr` package. If there are many channels, this may be specified as a vector, one element for each channel.
`quick`	`FALSE` repeats the detrending procedure (which has some inherent randomness) a few times to hone in on the best detrend. `TRUE` is quicker, performing the routine only once. `FALSE` is better.
`filt`	Do you want to smooth (`filt = 'mean'`) or median (`filt = 'median'`) filter the number image using `smooth_filter()` or `median_filter()` respectively? If selected, these are invoked here with a filter radius of 1 (with corners included, so each median is the median of 9 elements) and with the option `na_count = TRUE`. If you want to smooth/median filter the number image in a different way, first calculate the numbers without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result. If there are many channels, this may be specified as a vector, one element for each channel.
`s`	A positive number. The S-factor of microscope acquisition.
`offset`	Microscope acquisition parameters. See reference Dalal et al.
`readout_noise`	Microscope acquisition parameters. See reference Dalal et al.
`gamma`	Factor for correction of number n due to the illumination profile. The default (`gamma = 1`) has no effect. Changing gamma will have the effect of dividing the result by `gamma`, so the result with `gamma = 0.5` is two times the result with `gamma = 1`. For a Gaussian illumination profile, use `gamma = 0.3536`; for a Gaussian-Lorentzian illumination profile, use `gamma = 0.0760`.
`parallel`	Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

Note

Extreme number values (of magnitude greater than 3.40282e+38) will be written to the TIFF file as `NA`, since TIFF files cannot handle such huge numbers.

See Also

`number_timeseries()`

Examples

```
## Not run:
setwd(tempdir())
img <- ijtiff::read_tif(system.file("extdata", "50.tif", package = "nandb"))
ijtiff::write_tif(img, "img1.tif")
ijtiff::write_tif(img, "img2.tif")
number_timeseries_folder(def = "n", thresh = "Huang", frames_per_set = 20)

## End(Not run)
```

Index