Package 'materialmodifier'

October 13, 2022

Title Apply Photo Editing Effects

Version 1.1.0

Description
You can apply image processing effects that modifies the perceived material properties of objects in photos, such as gloss, smoothness, and blemishes. This is an implementation of the algorithm proposed by
Boyadzhiev et al. (2015) ``Band-Sifting Decomposition for Image Based Material Editing". Documentation and practical tips of the package is available at <https://github.com/tsuda16k/materialmodifier>.

URL https://github.com/tsuda16k/materialmodifier

BugReports https://github.com/tsuda16k/materialmodifier/issues/

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports jpeg, magrittr, methods, png, readbitmap, stringr, downloader, imager, moments

Depends R (>= 2.10)

NeedsCompilation no

Author Hiroyuki Tsuda [aut, cre] (<https://orcid.org/0000-0001-9396-5327>)

Maintainer Hiroyuki Tsuda <tsuda16k@gmail.com>

Repository CRAN

Date/Publication 2021-08-11 14:50:02 UTC

R topics documented:

`cimg2nimg` *cimg to nimg conversion*

Description

cimg to nimg conversion

Usage

```
cimg2nimg(im)
```

Arguments

`im`	a cimg object

Value

an nimg object

`face` *A face image.*

Description

A photograph obtained from a free stock photos site. pexels.com/photo/fashion-woman-cute-shoes-5704849/

Usage

```
face
```

Format

An array with 500 x 500 x 3 dimensions. Each dimension represents y-coordinate, x-coordinate, and color channel.

Examples

```
plot(face)
```

`get_BS_energy` *Calculate the BS feature energy*

Description

Calculate the BS feature energy

Usage

```
get_BS_energy(im, mask)
```

Arguments

`im`	An image.
`mask`	(optional) An image used for mask.

Value

a data frame

Examples

```
data = get_BS_energy(face)
```

`gf_decompose` *Scale-space decomposition by the guided filter*

Description

Scale-space decomposition by the guided filter

Usage

```
gf_decompose(im, log_epsilon = 1e-04, filter_epsilon = 0.01)
```

Arguments

`im`	an image
`log_epsilon`	offset for log transformation
`filter_epsilon`	epsilon parameter

Value

a list of images

gf_decompose_parts *Scale-space decomposition*

Description

Scale-space decomposition

Usage

```
gf_decompose_parts(dec)
```

Arguments

dec	output of gf_decompose_scale function

Value

a list of images

gf_decompose_scale *Scale-space decomposition by the guided filter*

Description

Scale-space decomposition by the guided filter

Usage

```
gf_decompose_scale(
  im,
  depth = NULL,
  log_epsilon = 1e-04,
  filter_epsilon = 0.01
)
```

Arguments

im	a grayscale image
depth	scale depth
log_epsilon	offset for log transformation
filter_epsilon	epsilon parameter

Value

a list of images

`gf_reconstruct` *Reconstruct the original image from decomposed data*

Description

Reconstruct the original image from decomposed data

Usage

```
gf_reconstruct(dec, scales, ind, include.residual = TRUE)
```

Arguments

`dec`	decomposed data
`scales`	which spatial scales to use for reconstruction
`ind`	a numeric vector
`include.residual`	either TRUE (default) or FALSE

Value

an image

`im_load` *Load image from file or URL*

Description

Load image from file or URL

Usage

```
im_load(file, name)
```

Arguments

`file`	path to file or URL
`name`	a string for name attribute. if missing, inferred from the file argument.

Value

an array of image data

Examples

```
## Not run:
# load an image from disk
im = im_load("path/to/your/image.jpg")
plot(im)

## End(Not run)
# load an image from URL
im = im_load("http://placehold.jp/150x150.png")
```

`im_save` *Save an image to disk*

Description

Save an image to disk

Usage

```
im_save(im, name, path, format = "png", quality = 0.95)
```

Arguments

`im`	An image.
`name`	Name of the image file.
`path`	The image is saved in this directory. For example, path = getwd()
`format`	Image format. Either "jpg", "png", "tiff", or "bmp". Default is "png".
`quality`	(jpg only) default is 0.95. Higher quality means less compression.

Value

No return value, called for side effects.

Examples

```
## Not run:
# face.png is saved to a path (if a path is specified)
im_save( face, path = NULL )
# img.png is saved to a path (if a path is specified)
im_save( face, name = "img", path = NULL )
# myimage.jpg is saved to a path (if a path is specified)
im_save( face, name = "myimage", path = NULL, format = "jpg" )

## End(Not run)
```

`modif` *Apply material editing effect*

Description

Apply material editing effect

Usage

```
modif(
  im,
  effect,
  strength,
  max_size = 1280,
  log_epsilon = 1e-04,
  filter_epsilon = 0.01
)
```

Arguments

`im`	An input image.
`effect`	A string naming the effect to apply. Either "gloss", "shine", "spots", "blemish", "rough", "stain", "shadow", or "aging".
`strength`	A numeric, which controls the strength of the effect. Strength values between 0 and 1 will reduce a feature, while strength values larger than 1 will boost a feature. A strength value of 1 does nothing. Negative values are allowed, which will invert a feature.
`max_size`	If the shorter side of the input image is larger than this value (the default is 1280), input image is resized before applying effects. Because the modif() function is very slow for large-resolution images, it is useful to limit the image resolution to speed-up the image processing. If you do not want to change the resolution of the input image, you can enter a large value for max_size, or set max_size = NA
`log_epsilon`	Offset for log transformation (default is 0.0001). Need not to change this value in most cases.
`filter_epsilon`	Epsilon parameter of the Guided filter (default is 0.01). Need not to change this value in most cases.

Value

an output image

Examples

```
plot(modif(face, effect = "shine", strength = 2.5)) # Apply the "shine" effect (make objects shiny)
plot(modif(face, effect = "shine", strength = 0.2)) # Less shiny effect with a parameter less than 1
plot(modif(face, effect = c("shine", "stain"), strength = c(0.2, 3))) # Less shiny and more stain
```

`modif2` *Apply material editing effect*

Description

Apply material editing effect

Usage

```
modif2(im, params, max_size = 1280, log_epsilon = 1e-04, filter_epsilon = 0.01)
```

Arguments

`im`	An input image.
`params`	A list of parameter values. Parameter names are freq, amp, sign, and strength.
`max_size`	If the shorter side of the input image is larger than this value (the default is 1280), input image is resized. The modif function is very slow for large-resolution images.
`log_epsilon`	Offset for log transformation (default is 0.0001). Need not to change this value in most cases.
`filter_epsilon`	Epsilon parameter of the Guided filter (default is 0.01). Need not to change this value in most cases.

Value

an output image

Examples

```
# shine effect
shine = list(freq = "H", amp = "H", sign = "P", strength = 2)
plot(modif2(face, params = shine))

# shine effect (equivalent to the above)
shine2 = list(freq = 1:4, amp = "H", sign = "P", strength = 2)
plot(modif2(face, params = shine2))

# you can specify a feature name directly, instead of specifying freq/amp/sign separately
plot( modif2( face, params = list( feature = "HHA", strength = 2 ) ) )
```

```
plot( modif2( face, params = list( feature = "1HP", strength = 3 ) ) )

# apply multiple effects at the same time
blemish = list(feature = "HLA", strength = 0.1) # less blemish
smooth = list(feature = "HHN", strength = 0.2) # smoother
plot(modif2(face, params = list(blemish, smooth)))
```

`modif_dim` *Check the scale information of an image*

Description

Check the scale information of an image

Usage

```
modif_dim(im)
```

Arguments

`im`	An image.

Value

A list of depth (number of scale subband images), indexes of high amplitude subbands, and indexes of low amplitude subbands.

Examples

```
modif_dim(face)
```

`nimg2cimg` *nimg to cimg conversion*

Description

nimg to cimg conversion

Usage

```
nimg2cimg(im)
```

Arguments

`im`	an nimg object

Value

a cimg object

`plot.nimg` *Display an image*

Description

Display an image

Usage

```
## S3 method for class 'nimg'
plot(x, rescale = FALSE, ...)
```

Arguments

`x`	an image
`rescale`	logical. if true, then pixel value is rescaled to range between 0 and 1.
`...`	other parameters to be passed to plot.default

Value

No return value, called for side effects.

Examples

```
plot(face)
```

Index