# Package 'redR'

September 3, 2018

**Type** Package

**Title** REgularization by Denoising (RED)

**Version** 1.0.1

**Maintainer** Adriano Passos <adriano.utfpr@gmail.com>

**Description** Regularization by Denoising uses a denoising engine to solve
many image reconstruction ill-posed inverse problems. This is a R
implementation of the algorithm developed by Romano et.al. (2016) <arXiv:1611.02862>. Currently,
only the gradient descent optimization framework is implemented. Also,
only the median filter is implemented as a denoiser engine. However,
(almost) any denoiser engine can be plugged in. There are currently available
3 reconstruction tasks: denoise, deblur and super-resolution. And again,
any other task can be easily plugged into the main function 'RED'.

**Depends** R (>= 3.4.0), imager

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1

**NeedsCompilation** no

**Author** Adriano Passos [aut, cre]

**Repository** CRAN

**Date/Publication** 2018-09-03 11:20:03 UTC

## R topics documented:

---

`cameraman` *Photograph of a cameraman*

---

## Description

This image is usually used as benchmark in SR problems

## Usage

```
cameraman
```

## Format

an image of class `cimg`

---

`degrade` *Degradation of an image*

---

## Description

This function degrades a high resolution image into a low resolution image.

## Usage

```
degrade(z, L = 1, s = cbind(0, 0), noise = 0, blur = 1, L1 = L,
  L2 = L)
```

## Arguments

| | |
|---|---|
| `z` | a `cimg`object containing the high resolution image |
| `L` | numeric indicating the overall scale change. This parameter will be override by L1 or L2 |
| `s` | numeric `p` by `2` matrix containing the registration parameters |
| `noise` | numeric indicating the standard deviation of the noise or an `cimg`object that will be added to the resampled z |
| `blur` | numeric indicating the blur range (for uniform blur) or an cimg object with the blur kernel to be convolved with z if nothing is provided an default kernel will be used. |
| `L1` | numeric indicating the directional scale change |
| `L2` | numeric indicating the directional scale change |

## Value

A degraded cimgobject

## Examples

```
degraded.lenna <- degrade(lenna, L = 4, noise = 0.05, blur = 3)
par(mfrow = c(1,2), mar = c(0,0,1,0)+0.1)
plot(lenna, axes = FALSE, interp = FALSE, main = 'Original Lenna')
plot(degraded.lenna, axes = FALSE, interp = FALSE, main = 'Degraded Lenna')
```

---

# error *Error measurements of images*

---

## Description

This function calculates error between two images

## Usage

```
MSE(x, y = NULL)

MAE(x, y = NULL)

PSNR(x, y)
```

## Arguments

| | |
|---|---|
| x, y | cimg objects |

## Functions

- MSE: Mean Squared Error
- MAE: Mean Absolute Error
- PSNR: Peak Signal-to-Noise Ratio

## Examples

```
degraded.lenna <- degrade(lenna, noise = 0.05)
MSE(lenna, degraded.lenna)
MAE(lenna, degraded.lenna)
PSNR(lenna, degraded.lenna)
#alternatively it can be done like:
MSE(lenna - degraded.lenna)
MAE(lenna - degraded.lenna)
```

## fft_convolve *Convolution of two images via FFT*

### Description

Convolution of two images via FFT

### Usage

```
fft_convolve(im, filter, deconvolution = FALSE)
```

### Arguments

| | |
|---|---|
| im, filter | cimg objects |
| deconvolution | logical indicating if the deconvolution should be performed |

### Examples

```
im <- lenna
filter <- imfill(9,9,val = 1)
blurred.im <- fft_convolve(im, filter)
deblurred.im <- fft_convolve(blurred.im, filter, deconvolution = TRUE)
par(mfrow = c(1,3), mar = c(0,0,1,0)+0.1)
plot(im, axes = FALSE, interp = FALSE, main = 'Original Lenna')
plot(blurred.im, axes = FALSE, interp = FALSE, main = 'Blurred Lenna')
plot(deblurred.im, axes = FALSE, interp = FALSE, main = 'deBlurred Lenna')
PSNR(im, blurred.im)
PSNR(im, deblurred.im)
```

## lenna *Photograph of Lenna*

### Description

The Lenna (or Lena) picture is one of the most widely used standard test images used for compression algorithms

### Usage

```
lenna
```

### Format

an image of class cimg

RED *RED: Regularization by Denoising*

## Description

RED: Regularization by Denoising

REgularization by Denoising

## Usage

```
RED(y, x0 = NULL, lambda = 1, sigma = 1, functional = "SR",
  engine = "MF", niter = 50, step = NULL, tol = 0.001, args = NULL)
```

## Arguments

| | |
|---|---|
| y | cimg object with the observed frame(s) |
| x0 | initial guess for the output image, if NULL an educated guess will be used. If a custom functional is provided this cant be NULL |
| lambda, sigma | numeric indicating the regularization parameters |
| functional | character with the optimization task or function with the functional to be used |
| engine | character indicating the denoised engine or function with the denoiser engine to be used |
| niter | numeric indicating the maximum number of iterations |
| step | numeric indicating the step size (if NULL an optimal step size will be used) |
| tol | numeric indicating the stopping criteria. The algorithm will stop when step < tol. Default = 0.001 |
| args | arguments to be passed implicitly to H HT and f |

## Examples

```
im <- lenna
y <- degrade(im, noise = 0.05)
x <- RED(y, sigma = 1, lambda = 5, functional = 'DN', niter = 50)
par(mfrow = c(1,2), mar = c(0,0,2,0)+0.1)
plot(y, interp = FALSE, axes = FALSE, main = 'Degraded im')
mtext(paste(round(PSNR(im, y),2), 'dB'), side = 1, line = -2)
plot(x, interp = FALSE, axes = FALSE, main = 'Restored im')
mtext(paste(round(PSNR(im, x),2), 'dB'), side = 1, line = -2)

## Not run:
im <- cameraman
y <- degrade(im, blur = 5)
y<- isoblur(im, 3, gaussian = TRUE)
x <- RED(y, sigma = 1, lambda = 4, functional = 'DB', niter = 1500)
par(mfrow = c(1,2), mar = c(0,0,2,0)+0.1)
```

```
plot(y, interp = FALSE, axes = FALSE, main = 'Degraded image')
mtext(paste(round(PSNR(im, y),2), 'dB'), side = 1, line = -2)
plot(x, interp = FALSE, axes = FALSE, main = 'Restored image')
mtext(paste(round(PSNR(im, x),2), 'dB'), side = 1, line = -2)

im <- cameraman
L = 2
s <- cbind(c(0,1,2,-2,1,3,-1,-3,-1), c(0,-1,2,1,-2,-3,3,-2,-3))
y <- degrade(im, L = L, s = s, noise = 0.05)
xref <- resize(imsplit(y,'z')[[1]], -100*L, -100*L, interpolation_type = 5)
x <- RED(y, sigma = 1, lambda = 5, functional = 'SR', niter = 50, args = list(scale = L, s=s))
par(mfrow = c(1,2), mar = c(0,0,2,0)+0.1)
plot(xref, interp = FALSE, axes = FALSE, main = 'Bicubic Interpolation')
mtext(paste(round(PSNR(im, xref),2), 'dB'), side = 1, line = -2)
plot(x, interp = FALSE, axes = FALSE, main = 'Super Resolved')
mtext(paste(round(PSNR(im, x),2), 'dB'), side = 1, line = -2)

im0 <- 0.2*pad(cameraman, 256, 'xy')
im1 <- lenna
im2 <- im1 - im0
y1 <- degrade(im1, noise = 0.05)
y2 <- degrade(im2, noise = 0.05)
y0 <- y1 - y2
x0 <- RED(y0, sigma = 1, lambda = 50, functional = 'DN', niter = 100)

par(mfrow = c(1,2), mar = c(0,0,2,0)+0.1)
plot(y0, interp = FALSE, axes = FALSE, main = 'naive')
mtext(paste(round(PSNR(im0, y0),2), 'dB'), side = 1, line = -2)
plot(x0, interp = FALSE, axes = FALSE, main = 'proposed')
mtext(paste(round(PSNR(im0, x0),2), 'dB'), side = 1, line = -2)

## End(Not run)
```

---

`register` *Registration parameter estimation*

---

## Description

Registration parameter estimation

## Usage

```
register(src, tar, method = "taylor", par0 = c(0, 0, 0), verbosity = 2,
  ...)
```

## Arguments

| | |
|---|---|
| `src, tar` | cimg objects |
| `method` | character indicating the method to be used |

| | |
|---|---|
| par0 | numeric vector for the initial guess for the registration parameters |
| verbosity | Numeric indicating the level of verbosity is displayed |
| ... | parameters to be passed to the optimization algorithm |

## Value

the registration parameters, usually a 2d vector.

## Examples

```
src <- cameraman

tar <- shift(cameraman, c(5,-15))
round(s <- register(src, tar, method = 'coarse', steps = 4), 4)

tar <- shift(cameraman, c(-1.155, 3.231))
round(s <- register(src, tar, method = 'taylor', tol = 1e-4), 4)

tar <- transform(cameraman, c(c(-1.155, 1.231, 0.121)))
round(s <- register(src, tar, method = 'taylor3', tol = 1e-4, maxiter = 100), 4)
```

---

# resample *Resampling of an image*

---

## Description

Resampling of an image

## Usage

```
resample(im, L = 1, L1 = L, L2 = L)
```

## Arguments

| | |
|---|---|
| im | cimg object |
| L | numeric indicating the overall scale change. This parameter will be override by L1 or L2 |
| L1, L2 | numeric indicating the directional scale change |

## Value

A resampled cimg object

## Examples

```
im <- lenna
par(mfrow = c(1,2), mar = rep(0,4)+0.1)
plot(im, axes = FALSE, interp = FALSE)
plot(resample(im, 1/4), axes = FALSE, interp = FALSE)
```

## shift *shifting operator*

### Description

shifting operator

### Usage

```
shift(im, s)
```

### Arguments

| | |
|---|---|
| im | cimg object |
| s | numeric p by 2 matrix containing the registration parameters |

### Value

shifted cimg object

### Examples

```
shift(cameraman, c(1,1))
shift(cameraman, cbind(c(1,1),c(-0.5,0.5)))
```

## transform *Transform an image*

### Description

Transform an image

### Usage

```
transform(im, s)
```

### Arguments

| | |
|---|---|
| im | cimg object |
| s | numeric 1 by 3 vector containing the registration parameters |

### Value

shifted cimg object

## Examples

```
shift(cameraman, c(1,1))
shift(cameraman, cbind(c(1,1),c(-0.5,0.5)))
```

# Index