Package ‘imageER’

April 12, 2024

Title Detect Edge in Images and Denoise (Recover) Images

Version 1.0.0

Description A packege for image processing, offering tools for edge detection and denoising applicable to both grayscale and color images. It is designed to analyze entire image or to slice image into smaller pieces for efficiency. Methods are described in: Peihua Qiu (1998) <doi:10.1214/aos/1024691468>, (2009) <doi:10.1007/s10463-007-0166-9>.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports pracma

R topics documented:

`detection_recover_blocking` *Function used to detect edge and denoise image with blocking capability*

Description

Detect edges and recover images in one step, efficiently handling non-square images by dividing them into smaller blocks to speed up calculations.

Usage

```
detection_recover_blocking(z, h, k, k1, block_size = 64)
```

Arguments

z	Input image, single channel only.
h	Parameter to adjust size of neighborhood during least square calculation, should be an integer.
k	Tuning parameter, size of neighborhood for edge calculation.
k1	Tuning parameter, size of neighborhood for denoising.
block_size	Size of blocks you wish to use, default size 64x64.

Value

A list of detected edges (an 0, 1 matrix of edge locations with 1 being edge location) and denoised image.

Examples

```
## Not run:
#Generate a surface with a circular jump in the middle
n <- 100
x <- y <- (1:n)/n
f <- matrix(0, n , n)

for (i in 1:n){
  for (j in 1:n){
    if ((x[i]-0.5)^2+(y[j]-0.5)^2 > 0.25^2){
      f[i,j] <- -2*(x[i]-0.5)^2-2*(y[j]-0.5)^2
    } else{
      f[i,j] <- -2*(x[i]-0.5)^2-2*(y[j]-0.5)^2+1
    }
  }
}

#Adding normal noise
sigma <- 1/2 * sd(as.vector(f))
set.seed(1234)
noise <- matrix(rnorm(n*n,0,sigma), n, n)
z <- f + noise

#Detection and recover
result = detection_recover_blocking(z, h=4, k=5, k1=5, block_size=64)

## End(Not run)
```

---

edge_detect *Function to detect edges/jumps on SQUARE image.*

Description

Detect edges/jumps of single channel square images (equal width and height).

Usage

```
edge_detect(z, k, h, alpha_n = 0.05)
```

Arguments

z	Input image, single channel only, square image only.
k	Tuning parameter, size of neighborhood for edge calculation, should be an odd integer.
h	Parameter to adjust size of neighborhood during least square calculation, should be an integer.
alpha_n	Parameter to adjust detection sensitivity, default 0.05.

Value

An 0, 1 matrix of edge locations with 1 being edge location

References

Qiu, Peihua. "Discontinuous Regression Surfaces Fitting." The Annals of Statistics 26, no. 6 (1998): 2218–45.

Examples

```
## Not run:
#Generate a surface with a circular jump in the middle
n <- 64
x <- y <- (1:n)/n
f <- matrix(0, n , n)

for (i in 1:n){
  for (j in 1:n){
    if ((x[i]-0.5)^2+(y[j]-0.5)^2 > 0.25^2){
      f[i,j] <- -2*(x[i]-0.5)^2-2*(y[j]-0.5)^2
    } else{
      f[i,j] <- -2*(x[i]-0.5)^2-2*(y[j]-0.5)^2+1
    }
  }
}

#Adding normal noise
sigma <- 1/2 * sd(as.vector(f))
set.seed(1234)
noise <- matrix(rnorm(n*n,0,sigma), n, n)
z <- f + noise

#Detecting Edge
edge_map <- edge_detect(z, k=5, h=4, alpha_n=0.05)

## End(Not run)
```

`jump_preserving_estimator`

Generate a jump-preserving surface estimator from noisy data.

Description

Generate a surface estimator from noisy data with jump information well preserved. This function is primarily used in `par_select()` to generate bootstrap sample for Hasudorff Distance estimation.

Usage

```
jump_preserving_estimator(z, h, sigma_hat, alpha = 0.05)
```

Arguments

`z`	Input image, single channel only, square image only.
`h`	Parameter to adjust size of neighborhood for local surface estimation, should be an integer.
`sigma_hat`	Estimated noise level (noise variance) of input surface
`alpha`	Significance level to control the reconstruction performance of jump, default 0.05. Can be chosen from (0.5, 0.4, 0.3, 0.2, 0.15, 0.1, 0.075, 0.05, 0.025, 0.01, 0.001, 0.0001)

Value

Estimated surface.

References

Qiu, Peihua. "The Local Piecewisely Linear Kernel Smoothing Procedure for Fitting Jump Regression Surfaces." Technometrics 46, no. 1 (February 1, 2004): 87–98. https://doi.org/10.1198/004017004000000149.

Qiu, Peihua. "Jump-Preserving Surface Reconstruction from Noisy Data." Annals of the Institute of Statistical Mathematics 61, no. 3 (September 1, 2009): 715–51. https://doi.org/10.1007/s10463-007-0166-9.

Examples

```
## Not run:
#Generate a surface with a circular jump in the middle
n <- 64
x <- y <- (1:n)/n
f <- matrix(0, n , n)

for (i in 1:n){
  for (j in 1:n){
    if ((x[i]-0.5)^2+(y[j]-0.5)^2 > 0.25^2){
      f[i,j] <- -2*(x[i]-0.5)^2-2*(y[j]-0.5)^2
    } else{
      f[i,j] <- -2*(x[i]-0.5)^2-2*(y[j]-0.5)^2+1
    }
  }
}

#Adding normal noise
sigma <- 1/2 * sd(as.vector(f))
set.seed(1234)
noise <- matrix(rnorm(n*n,0,sigma), n, n)
z <- f + noise
```

```
#Estimate sigma_hat(this is just an example, there are other ways to estimate sigma_hat)
f_hat_int <- jump_preserving_estimator(z, h=4, sigma_hat=0, alpha=0.05)
sigma_est <- sqrt(sum((z-f_hat_int)^2)/n^2)
f_hat_upd <- f_hat_upd <- jump_preserving_estimator(z, h=4 ,sigma_hat=sigma_est, alpha=0.05)

## End(Not run)
```

`par_select` *Function to select optimal tuning parameter for edge detection (k) and surface recovery (k1)*

Description

Choose optimal tuning parameter for jump/edge detection and surface recovery/denoising. Omega value can be adjusted for better edge detection or better denoising.

Usage

```
par_select(k, k1, z, h, alpha_n = 0.05, alpha_jp = 0.05, omega = 0.5)
```

Arguments

`k`	A vector of tuning parameter k, should all be odd numbers.
`k1`	A vector of tuning parameter k1.
`z`	Input image, single channel only, square image only.
`h`	Parameter to adjust size of neighborhood during least square calculation, should be an integer.
`alpha_n`	Parameter to adjust detection sensitivity, default 0.05.
`alpha_jp`	Significance level to control the reconstruction performance of jump, default 0.05. Can be chosen from (0.5, 0.4, 0.3, 0.2, 0.15, 0.1, 0.075, 0.05, 0.025, 0.01, 0.001, 0.0001)
`omega`	Parameter to adjust for selection preference, default 0.5. If greater than 0.5, focus more on edge detection, if less than 0.5, focus more on surface recovery.

Value

A message of selection.

Examples

```
## Not run:
# Generate a surface with a circular jump in the middle
n <- 64
x <- y <- (1:n) / n
f <- matrix(0, n, n)

for (i in 1:n) {
  for (j in 1:n) {
    if ((x[i] - 0.5)^2 + (y[j] - 0.5)^2 > 0.25^2) {
      f[i, j] <- -2 * (x[i] - 0.5)^2 - 2 * (y[j] - 0.5)^2
    } else {
      f[i, j] <- -2 * (x[i] - 0.5)^2 - 2 * (y[j] - 0.5)^2 + 1
    }
  }
}

# Adding normal noise
sigma <- 1 / 2 * sd(as.vector(f))
set.seed(1234)
noise <- matrix(rnorm(n * n, 0, sigma), n, n)
z <- f + noise

k <- c(3, 5, 7, 9, 11)
k1 <- c(5, 10, 15, 20, 25)

par_select(k, k1, z, h = 3, alpha_n = 0.05, alpha_jp = 0.05, omega = 0.5)

## End(Not run)
```

`recover_surface` *Function to recover noisy surface with jump detail preserved.*

Description

Denoise single channel square images, jump/edge information will be retained instead of being blurred. Will need edge information (output of `edge_detect()`) as input.

Usage

```
recover_surface(z, k1, edge)
```

Arguments

`z`	Input image, single channel only, square image only.
`k1`	Tuning parameter, size of neighborhood for local weighted average calculation.
`edge`	A matrix of detected edges, should be ourput from `edge_detect`.

Value

A denoised image.

References

Qiu, Peihua. "Discontinuous Regression Surfaces Fitting." The Annals of Statistics 26, no. 6 (1998): 2218–45.

Examples

```
## Not run:
#Generate a surface with a circular jump in the middle
n <- 64
x <- y <- (1:n)/n
f <- matrix(0, n , n)
```

```
for (i in 1:n){
  for (j in 1:n){
    if ((x[i]-0.5)^2+(y[j]-0.5)^2 > 0.25^2){
      f[i,j] <- -2*(x[i]-0.5)^2-2*(y[j]-0.5)^2
    } else{
      f[i,j] <- -2*(x[i]-0.5)^2-2*(y[j]-0.5)^2+1
    }
  }
}

#Adding normal noise
sigma <- 1/2 * sd(as.vector(f))
set.seed(1234)
noise <- matrix(rnorm(n*n,0,sigma), n, n)
z <- f + noise

#Detecting Edge
edge_map <- edge_detect(z, k=5, h=4, alpha_n=0.05)

#Recover/Denoise Surface
recover_surface(z, k1=5, edge=edge_map)

## End(Not run)
```

Index