Package ‘DRIP’

October 7, 2018

Version 1.4

Date 2018-10-06

Title Discontinuous Regression and Image Processing

Author Yicheng Kang <kangx276@umn.edu>

Maintainer Yicheng Kang <kangx276@umn.edu>

Description This is a collection of functions for discontinuous regression analysis and image processing (DRIP). A recent addition is the blind image deblurring via jump-preserving extrapolation.

Depends R (>= 2.10.0), readbitmap, graphics, stats

License GPL (>= 3)

Encoding UTF-8

LazyData true

NeedsCompilation yes

SystemRequirements Gnu Scientific Library version >= 1.12

Repository CRAN

Date/Publication 2018-10-07 04:10:37 UTC

R topics documented:

`brain` *Brain image*

Description

This file contains data of a brain image. It has 217x217 pixels. Gray levels are in the range [0, 255]. In the data file, observations are listed as a 217 x217 matrix. This image has blur involved.

Usage

```
brain
```

Format

This dataset is saved in an ascii file.

circles — *Image of circles*

Description

This file contains data of the original circles image. It has 256x256 pixels. Gray levels are in the range [0, 255]. In the data file, observations are listed as a 256x256 matrix.

Usage

```
circles
```

Format

This dataset is saved in an ascii file.

cv.jpex — *Bandwidth Selection and Noise Level Estimation*

Description

cv.jpex() selects the leave-one-out cross validation (CV) bandwidth for LLK smoothing and estimates the noise level in the input image. Both the bandwidth parameter and the noise level are required inputs for the blind image deblurring procedure jpex().

Usage

```
cv.jpex(image, bandwidth)
```

Arguments

`image`	A blurry image to deblurred
`bandwidth`	A vector of positive integers that specifies the size of the neighborhood for local smoothing.

Value

`LLK`	The estimated surface by local linear kernel (LLK) smoothing, using the CV selected bandwidth.
`sigma`	The estimated noise level, defined as the square root of the mean squared error (MSE) between LLK and the input image.
`cv`	A vector of the same length as that of bandwidth. Each element in the vector is the leave-one-out CV error associated with the corresponding bandwidth parameter.
`bandwidth`	The bandwidth parameters input by user.
`band.min`	The bandwidth parameter that results in the smallest CV error.

Author(s)

Yicheng Kang

References

Kang, Y. (2018), “Consistent Blind Image Deblurring Using Jump-Preserving Extrapolation”.

See Also

jpex

Examples

```
require(DRIP)
data(stopsign)
out = cv.jpex(stopsign, c(2,3))
```

---

diffLC2K *local constant kernel difference*

Description

Compute difference between two one-sided LC2K estimators along the gradient direction.

Usage

```
diffLC2K(image, bandwidth, plot)
```

Arguments

image	A square matrix object of size n by n, no missing value allowed.
bandwidth	A positive integer to specify the number of pixels used in the local smoothing.
plot	If plot = TRUE, an image of the difference at each pixel is plotted.

Details

At each pixel, the gradient is estimated by a local linear kernel smoothing procedure. Next, the local neighborhood is divided into two halves along the direction perpendicular to $(\widehat{f}'_x, \widehat{f}'_y)$. Then the one-sided deblurring local constant kernel (LC2K) estimates are obtained in the two half neighborhoods respectively.

Value

Returns a matrix of the estimated difference, $|\widehat{f}_+ - \widehat{f}_-|$, at each pixel.

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

See Also

diffLCK, diffLLK, diffLL2K, stepEdgeLC2K

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
diff = diffLC2K(image = sar, bandwidth = 4)
```

---

diffLCK — *local constant kernel difference*

Description

Compute difference between two one-sided LCK estimators along the gradient direction.

Usage

```
diffLCK(image, bandwidth, plot)
```

Arguments

image	A square matrix object of size n by n, no missing value allowed.
bandwidth	A positive integer to specify the number of pixels used in the local smoothing.
plot	If plot = TRUE, an image of the difference at each pixel is plotted.

Details

At each pixel, the gradient is estimated by a local linear kernel smoothing procedure. Next, the local neighborhood is divided into two halves along the direction perpendicular to $(\widehat{f}'_x, \widehat{f}'_y)$. Then the one- sided local constant kernel (LCK) estimates are obtained in the two half neighborhoods respectively.

Value

Returns a matrix of the estimated difference, $|\widehat{f}_+ - \widehat{f}_-|$, at each pixel.

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

See Also

diffLLK, diffLC2K, diffLL2K, stepEdgeLCK

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
diff = diffLCK(image = sar, bandwidth = 4)
```

---

diffLL2K *local linear kernel difference*

Description

Compute difference between two one-sided LL2K estimators along the gradient direction.

Usage

```
diffLL2K(image, bandwidth, plot)
```

Arguments

image	A square matrix object of size n by n, no missing value allowed.
bandwidth	A positive integer to specify the number of pixels used in the local smoothing.
plot	If plot = TRUE, an image of the difference at each pixel is plotted.

Details

At each pixel, the gradient is estimated by a local linear kernel smoothing procedure. Next, the local neighborhood is divided into two halves along the direction perpendicular to $(\widehat{f}'_x, \widehat{f}'_y)$. Then the one-sided deblurring local linear kernel (LL2K) estimates are obtained in the two half neighborhoods respectively.

Value

Returns a matrix of the estimated difference, $|\widehat{f}_+ - \widehat{f}_-|$, at each pixel.

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

See Also

diffLCK, diffLC2K, diffLLK, stepEdgeLL2K

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
diff = diffLL2K(image = sar, bandwidth = 6)
```

`diffLLK` *local linear kernel difference*

Description

Compute difference between two one-sided LLK estimators along the gradient direction.

Usage

```
diffLLK(image, bandwidth, plot)
```

Arguments

`image`	A square matrix object of size n by n, no missing value allowed.
`bandwidth`	A positive integer to specify the number of pixels used in the local smoothing.
`plot`	If plot = TRUE, an image of the difference at each pixel is plotted.

Details

At each pixel, the gradient is estimated by a local linear kernel smoothing procedure. Next, the local neighborhood is divided into two halves along the direction perpendicular to $(\widehat{f}'_x, \widehat{f}'_y)$. Then the one-sided local linear kernel (LLK) estimates are obtained in the two half neighborhoods respectively.

Value

Returns a matrix of the estimated difference, $|\widehat{f}_+ - \widehat{f}_-|$, at each pixel.

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

See Also

`diffLCK`, `diffLC2K`, `diffLL2K`, `stepEdgeLLK`

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
diff = diffLLK(image = sar, bandwidth = 6)
```

dKQ *edge detection, performance measure*

Description

Compute the d_KQ distance between two sets of edge pixels. It can be used as a performance measure for (step/roof) edge detectors

Usage

```
dKQ(edge1, edge2)
```

Arguments

edge1	One set of pixels
edge2	The other set of pixels

Details

The mathematical definition of d_{KQ} is as follows. $d_{KQ}(S_1, S_2) = \frac{0.5}{|S_1|} \sum_{p_1 \in S_1} d_E(p_1, S_2) + \frac{0.5}{|S_2|} \sum_{p_2 \in S_2} d_E(p_2, S_1)$, where S_1 and S_2 are two point sets, and d_E denotes the Euclidean distance.

Value

Value of the d_{KQ}

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

Examples

```
mat1 = matrix(c(1, rep(0, 3)), ncol = 2)
mat2 = matrix(c(rep(0, 3), 1), ncol = 2)
dKQ(mat1, mat2)
```

jpex *Blind Image Deblurring*

Description

jpex() takes in any square matrix (noisy blurry image) and deblurs it.

Usage

```
jpex(image, bandwidth, alpha, sigma)
```

Arguments

`image`	An input blurry image to deblurred. The input image is represented a square matrix.
`bandwidth`	A positive integer that specifies the size of the neighborhood for local smoothing.
`alpha`	A numberic between 0 and 1. This is the signifance level for the Chi-square hypothesis test that a given pixel is in a continuity region and not affected by the blur.
`sigma`	A positive numeric for the noise level in the blurred image. It is used in the Chi-square test.

Value

`deblurred`	The deblurred image
`edge`	The square matrix, the element of which is the value of the Chi-square test statistic at the pixel location. One can classify a given pixel as a blurry pixcel if edge[i,j]>qchisq(1-alpha, 2).

Author(s)

Yicheng Kang

References

Kang, Y. (2018) “Consistent Blind Image Deblurring Using Jump-Preserving Extrapolation”.

See Also

cv.jpex

Examples

```
require(DRIP)
data(stopsign)
out = jpex(image=stopsign, bandwidth=as.integer(2), sigma=0.00623, alpha=0.001)
```

JPLLK_surface *Denoising and jump-preserving*

Description

Estimate surface using piecewise local linear kernel smoothing. Bandwidth is chosen by leave-one-out cross validation.

Usage

```
JPLLK_surface(image, bandwidth, plot = FALSE)
```

Arguments

image	A square matrix object of size n by n, no missing value allowed.
bandwidth	A numeric vector with positive integers, which specify the number of pixels used in the local smoothing. The final fitted surface chooses the optimal bandwidth from those provided by users.
plot	If plot = TRUE, the image of the fitted surface is plotted

Details

At each pixel, the gradient is estimated by a local linear kernel smoothing procedure. Next, the local neighborhood is divided into two halves along the direction perpendicular to $(\widehat{f}'_x, \widehat{f}'_y)$. Then the one-sided local linear kernel (LLK) estimates are obtained in the two half neighborhoods respectively. Among these two one-sided estimates, the one with smaller weighted mean square error is chosen to be the final estimate of the regression surface at the pixel.

Value

A list of fitted values, residuals, chosen bandwidth and estimated sigma.

References

Qiu, P., "Jump-preserving surface reconstruction from noisy data", *Annals of the Institute of Statistical Mathematics*, **61(3)**, 2009, 715-751.

See Also

threeStage, surfaceCluster

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
fit = JPLLK_surface(image=sar, bandwidth=c(3, 4))
```

`kid` *Image of a kid taking test*

Description

This file contains data of original kid image. The image has 387x387 pixels. Gray levels are in the range [0, 255]. In the data file, observations are listed as a 387x387 matrix. This image has spatially variant blur involved.

Usage

```
kid
```

Format

This dataset is saved in an ascii file.

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

`lena` *Image of Lena*

Description

This file contains data of the original Lena image. It has 512x512 pixels. Gray levels are in the range [0, 255]. In the data file, observations are listed as a 512x512 matrix.

Usage

```
lena
```

Format

This dataset is saved in an ascii file.

References

November 1972 issue of Playboy magazine

modify1 *Edge detection, post processing*

Description

Modify detected edges to make them thin.

Usage

```
modify1(bandwidth, image, edge, plot)
```

Arguments

image	A matrix that represents the image.
bandwidth	A positive integer to specify the number of pixels used in the local smoothing.
edge	A matrix of 0 and 1 represents detected edge pixels.
plot	If plot = TRUE, images of detected edges before the modification and after the modification are plotted.

Details

A local-smoothing based edge detection algorithm may flag deceptive edge pixel candidates. One kind of such candidates consists of those close to the real edges. They occur due to the nature of local smoothing. That is, if the point (x_i, y_j) is flagged, then its neighboring pixels will be flagged with high probability. This kind of deceptive candidates can make the detected edges thick. This modification procedure makes the detected edges thin.

Value

Returns a matrix of zeros and ones of the same size as edge.

References

Qiu, P. and Yandell, B., "Jump detection in regression surfaces," *Journal of Computational and Graphical Statistics* **6(3)**, 1997, 332-354.

See Also

modify2

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
edge = stepEdgeLCK(sar, 4, 20)
out = modify1(4, sar, edge)
```

modify2 *Edge detection, post processing*

Description

Delete deceptive edge pixels that are scattered in the design space.

Usage

```
modify2(bandwidth, edge, plot)
```

Arguments

bandwidth	A positive integer to specify the number of pixels used in the local smoothing.
edge	A matrix of 0 and 1 represents detected edge pixels.
plot	If plot = TRUE, images of detected edges before the modification and after the modification are plotted.

Details

A local-smoothing based edge detection algorithm may flag deceptive edge pixel candidates. One kind of such candidates existis due to the nature of hypothesis testing, on which the threshold value of the edge detection criterion is based. That is, a point (x_i, y_j) could be flagged as a edge pixel with certain probability, even if it is actually not a edge pixel. Deceptive candidates of this kind are scattered in the whole design space. This modification procedure deletes scattered edge pixel candidates.

Value

Returns a matrix of zeros and ones of the same size as edge.

References

Qiu, P. and Yandell, B., "Jump detection in regression surfaces," *Journal of Computational and Graphical Statistics* **6(3)**, 1997, 332-354.

See Also

modify1

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
edge = stepEdgeLCK(sar, 4, 20)
out = modify2(4, edge)
```

peppers *Image of peppers*

Description

This file contains data of the original peppers image. It has 512x512 pixels. Gray levels are in the range [0, 255]. In the data file, observations are listed as a 512x512 matrix.

Usage

```
peppers
```

Format

This dataset is saved in an ascii file.

roofDiff *roof/valley edge detection*

Description

Compute difference between two one-sided gradient estimators.

Usage

```
roofDiff(image, bandwidth, blur)
```

Arguments

`image`	A square matrix object of size n by n, no missing value allowed.
`bandwidth`	A positive integer to specify the number of pixels used in the local smoothing.
`blur`	If blur = TRUE, besides the conventional 2-D kernel function, a univariate kernel function is used to address the issue of blur.

Details

At each pixel, the second-order derivarives (i.e., f''_{xx}, f''_{xy}, and f''_{yy}) are estimated by a local quadratic kernel smoothing procedure. Next, the local neighborhood is first divided into two halves along the direction perpendicular to $(\widehat{f}''_{xx}, \widehat{f}''_{xy})$. Then the one-sided estimates of f'_{x+} and f'_{x-} are obtained respectively by local linear kernel smoothing. The estimates of f'_{y+} and f'_{y-} are obtained by the same procedure except that the neighborhood is divided along the direction $(\widehat{f}''_{xy}, \widehat{f}''_{yy})$.

Value

Returns a matrix where each entry is the maximum of the differences: $|\widehat{f}_{x+} - \widehat{f}_{x-}|$ and $|\widehat{f}_{y+} - \widehat{f}_{y-}|$ at each pixel.

References

Qiu, P., and Kang, Y. "Blind Image Deblurring Using Jump Regression Analysis," *Statistica Sinica*, **25**, 2015, 879-899.

See Also

`roofEdgeParSel`, `roofEdge`

Examples

```
data(peppers)
#diff = roofDiff(image = peppers, bandwidth = 8) # Time consuming
```

---

roofEdge *Edge detection, denoising and deblurring*

Description

Detect roof/valley edges in an image using piecewise local linear kernel smoothing.

Usage

```
roofEdge(image, bandwidth, thresh, edge1, blur, plot)
```

Arguments

`image`	A square matrix object of size n by n, no missing value allowed.
`bandwidth`	A positive integer to specify the number of pixels used in the local smoothing.
`thresh`	Threshold value used in the edge detection criterion.
`edge1`	Step edges. The function excludes step edges when detects roof/valley edges.
`blur`	If blur = TRUE, besides the conventional 2-D kernel function, a univariate kernel function is used in the local smoothing to address the issue of blur.
`plot`	If plot = TRUE, an image of detected edges is plotted.

Details

At each pixel, the second-order derivarives (i.e., f''_{xx}, f''_{xy}, and f''_{yy}) are estimated by a local quadratic kernel smoothing procedure. Next, the local neighborhood is first divided into two halves along the direction perpendicular to ($\widehat{f}''_{xx}$, $\widehat{f}''_{xy}$). Then the one-sided estimates of f'_{x+} and f'_{x-} are obtained respectively by local linear kernel smoothing. The estimates of f'_{y+} and f'_{y-} are obtained by the same procedure except that the neighborhood is divided along the direction ($\widehat{f}''_{xy}$, $\widehat{f}''_{yy}$). The pixel is flagged as a roof/valley edge pixel if $max(|\widehat{f}_{x+} - \widehat{f}_{x-}|, |\widehat{f}_{y+} - \widehat{f}_{y-}|) >$ the specified thresh and there is no step edge pixels in the neighborhood.

Value

Returns a matrix of zeros and ones of the same size as image.

References

Qiu, P., and Kang, Y. "Blind Image Deblurring Using Jump Regression Analysis," *Statistica Sinica*, **25**, 2015, 879-899.

See Also

roofEdgeParSel, roofDiff

Examples

```
data(peppers)
# Not run
#step.edges = stepEdgeLLK(peppers, bandwidth=6, thresh=25, plot=FALSE)
#roof.edges = roofEdge(image=peppers, bandwidth=9, thresh=3000, edge1=step.edges,
#     blur=FALSE, plot=FALSE) # Time consuming
#edges = step.edges + roof.edges
#par(mfrow=c(2,2))
#image(1-step.edges, col=gray(0:1))
#image(1-roof.edges, col=gray(0:1))
#image(1-edges, col=gray(0:1))
#image(peppers, col=gray(c(0:255)/255))
```

---

roofEdgeParSel *roof/valley edge detection, parameter selection*

Description

Select bandwidth and threshold value for the roof/valley edge detector using bootstrap procedure

Usage

```
roofEdgeParSel(image, bandwidth, thresh, nboot, edge1, blur=FALSE)
```

Arguments

image	A square matrix object of size n by n, no missing value allowed.
bandwidth	Positive integers to specify the number of pixels used in the local smoothing. These are the bandwidth parameters to be chosen from.
thresh	Threshold values to be chosen from.
nboot	Number of bootstrap samples.
edge1	Step edges. The function excludes step edges when detect roof/valley edges.
blur	TRUE if the image contains blur, FALSE otherwise.

Details

If *blur=TRUE*, then a conventional local linear kernel smoothing is applied to estimate the blurred surface; Bootstrap samples are obtained by drawing with replacement from the residuals and the d_{KQ} is computed for the detected edges of the original sample and those of the bootstrap samples. If *blur=FALSE*, the procedure is the same as when *blur=TRUE* except that a jump-preserving kernel smoothing procedure is used to obtain residuals.

Value

Returns a list of the selected bandwdith, the selected threshold value, and a matrix of d_{KQ} values with each entry corresponding to each combination of bandwdith and threshold.

References

Qiu, P., and Kang, Y. "Blind Image Deblurring Using Jump Regression Analysis," *Statistica Sinica*, **25**, 2015, 879-899.

See Also

roofDiff, roofEdge

Examples

```
data(peppers) # Peppers image is bundled with the package and it is a
              # standard test image in image processing literature.
#step.edges = stepEdgeLLK(image=peppers, bandwidth=9, thresh=17, plot=FALSE)
#parSel = roofEdgeParSel(image=peppers, bandwidth=5:6, thresh=
#c(5000,10000), nboot=10, edge1=step.edges, blur=FALSE) # Time Consuming
```

---

sar — *Synthetic aperture radar image of an area near Thetford forest, England*

Description

This file contains data of original sar image. The image has 250x250 pixels. Gray levels are in the range [0, 255]. In the data file, observations are listed as a 250x250 matrix. This image contains much noise.

Usage

```
sar
```

Format

This dataset is saved in an ascii file.

References

This data can be downloaded from: http://peipa.essex.ac.uk/ipa/pix/books/glasbey-horgan/

`stepEdgeLC2K` *Edge detection, denoising and deblurring*

Description

Detect step edges in an image using piecewise local constant kernel smoothing.

Usage

```
stepEdgeLC2K(image, bandwidth, thresh, plot)
```

Arguments

`image`	A square matrix object of size n by n, no missing value allowed.
`bandwidth`	A positive integer to specify the number of pixels used in the local smoothing.
`thresh`	Threshold value used in the edge detection criterion.
`plot`	If plot = TRUE, an image of detected edges is plotted.

Details

At each pixel, the gradient is estimated by a local linear kernel smoothing procedure. Next, the local neighborhood is divided into two halves along the direction perpendicular to $(\widehat{f}'_x, \widehat{f}'_y)$. Then the one-sided deblurring local constant kernel (LC2K) estimates are obtained in the two half neighborhoods respectively. The pixel is flagged as a step edge pixel if $|\widehat{f}_+ - \widehat{f}_-| > u$, where u is a threshold value.

Value

Returns a matrix of zeros and ones of the same size as image. Value one represent edge pixels and value zero represent non-edge pixels.

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

See Also

`stepEdgeLCK`, `stepEdgeLLK`, `stepEdgeLL2K`, `diffLC2K`

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
edge = stepEdgeLC2K(image = sar, bandwidth = 4, thresh = 20)
```

stepEdgeLCK — *Edge detection, denoising and deblurring*

Description

Detect step edges in an image using piecewise local constant kernel smoothing.

Usage

```
stepEdgeLCK(image, bandwidth, thresh, plot)
```

Arguments

`image`	A square matrix object of size n by n, no missing value allowed.
`bandwidth`	A positive integer to specify the number of pixels used in the local smoothing.
`thresh`	Threshold value used in the edge detection criterion.
`plot`	If plot = TRUE, an image of detected edges is plotted.

Details

At each pixel, the gradient is estimated by a local linear kernel smoothing procedure. Next, the local neighborhood is divided into two halves along the direction perpendicular to $(\widehat{f}'_x, \widehat{f}'_y)$. Then the one- sided local constant kernel (LCK) estimates are obtained in the two half neighborhoods respectively. The pixel is flagged as a step edge pixel if $|\widehat{f}_+ - \widehat{f}_-| > u$, where u is a threshold value.

Value

Returns a matrix of zeros and ones of the same size as image. Value one represent edge pixels and value zero represent non-edge pixels.

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

See Also

`stepEdgeLC2K`, `stepEdgeLLK`, `stepEdgeLL2K`, `diffLCK`

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
edge = stepEdgeLCK(image = sar, bandwidth = 4, thresh = 20)
```

stepEdgeLL2K *Edge detection, denoising and deblurring*

Description

Detect step edges in an image using piecewise local linear kernel smoothing.

Usage

```
stepEdgeLL2K(image, bandwidth, thresh, plot)
```

Arguments

`image`	A square matrix object of size n by n, no missing value allowed.
`bandwidth`	A positive integer to specify the number of pixels used in the local smoothing.
`thresh`	Threshold value used in the edge detection criterion.
`plot`	If plot = TRUE, an image of detected edges is plotted.

Details

At each pixel, the gradient is estimated by a local linear kernel smoothing procedure. Next, the local neighborhood is divided into two halves along the direction perpendicular to $(\widehat{f}'_x, \widehat{f}'_y)$. Then the one-sided deblurring local linear kernel (LL2K) estimates are obtained in the two half neighborhoods respectively. The pixel is flagged as a step edge pixel if $|\widehat{f}_+ - \widehat{f}_-| > u$, where u is a threshold value.

Value

Returns a matrix of zeros and ones of the same size as image. Value one represent edge pixels and value zero represent non-edge pixels.

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

See Also

`stepEdgeLCK`, `stepEdgeLLK`, `stepEdgeLC2K`, `diffLL2K`

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
edge = stepEdgeLL2K(image = sar, bandwidth = 6, thresh = 20)
```

stepEdgeLLK *Edge detection, denoising and deblurring*

Description

Detect step edges in an image using piecewise local linear kernel smoothing.

Usage

```
stepEdgeLLK(image, bandwidth, thresh, plot)
```

Arguments

`image`	A square matrix object of size n by n, no missing value allowed.
`bandwidth`	A positive integer to specify the number of pixels used in the local smoothing.
`thresh`	Threshold value used in the edge detection criterion.
`plot`	If plot = TRUE, an image of detected edges is plotted.

Details

At each pixel, the gradient is estimated by a local linear kernel smoothing procedure. Next, the local neighborhood is divided into two halves along the direction perpendicular to $(\widehat{f}'_x, \widehat{f}'_y)$. Then the one-sided local linear kernel (LLK) estimates are obtained in the two half neighborhoods respectively. The pixel is flagged as a step edge pixel if $|\widehat{f}_+ - \widehat{f}_-| > u$, where u is a threshold value.

Value

Returns a matrix of zeros and ones of the same size as image. Value one represent edge pixels and value zero represent non-edge pixels.

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

See Also

`stepEdgeLCK`, `stepEdgeLC2K`, `stepEdgeLL2K`, `diffLLK`

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
edge = stepEdgeLLK(image = sar, bandwidth = 9, thresh = 17)
```

stepEdgeParSelLC2K *edge detection, parameter selection*

Description

Select bandwidth and threshold value for LC2K edge detector using bootstrap procedure

Usage

```
stepEdgeParSelLC2K(image, bandwidth, thresh, nboot)
```

Arguments

image	A square matrix object of size n by n, no missing value allowed.
bandwidth	Positive integers to specify the number of pixels used in the local smoothing. These are the bandwidth parameters to be chosen from.
thresh	Threshold values to be chosen from.
nboot	Number of bootstrap samples.

Details

A jump-preserving local linear kernel smoothing is applied to estimate the discontinuous regression surface; Bootstrap samples are obtained by drawing with replacement from the residuals and the d_{KQ} is computed for the detected edges of the original sample and those of the bootstrap samples.

Value

Returns a list of the selected bandwdith, the selected threshold value, and a matrix of d_{KQ} values with each entry corresponding to each combination of bandwdith and threshold.

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

See Also

stepEdgeParSelLCK, stepEdgeParSelLLK, stepEdgeParSelLL2K, stepEdgeLC2K

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
#parSel = stepEdgeParSelLC2K(image=sar, bandwidth=4:5, thresh=
#19:20, nboot=10) # Time consuming
```

stepEdgeParSelLCK *edge detection, parameter selection*

Description

Select bandwidth and threshold value for LCK edge detector using bootstrap procedure

Usage

```
stepEdgeParSelLCK(image, bandwidth, thresh, nboot)
```

Arguments

`image`	A square matrix object of size n by n, no missing value allowed.
`bandwidth`	Positive integers to specify the number of pixels used in the local smoothing. These are bandwidth parameters to be chosen from.
`thresh`	Threshold values to be chosen from.
`nboot`	Number of bootstrap samples.

Details

A jump-preserving local linear kernel smoothing is applied to estimate the discontinuous regression surface; Bootstrap samples are obtained by drawing with replacement from the residuals and the d_{KQ} is computed for the detected edges of the original sample and those of the bootstrap samples.

Value

Returns a list of the selected bandwdith, the selected threshold value, and a matrix of d_{KQ} values with each entry corresponding to each combination of bandwdith and threshold.

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

See Also

`stepEdgeParSelLC2K`, `stepEdgeParSelLLK`, `stepEdgeParSelLL2K`, `stepEdgeLCK`

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
#parSel = stepEdgeParSelLCK(image=sar, bandwidth=4:5, thresh=
#19:20, nboot=10) # Time consuming
```

stepEdgeParSelLL2K *edge detection, parameter selection*

Description

Select threshold value for LL2K edge detector using bootstrap procedure

Usage

```
stepEdgeParSelLL2K(image, bandwidth, thresh, nboot)
```

Arguments

image	A square matrix object of size n by n, no missing value allowed.
bandwidth	Positive integers to specify the number of pixels used in the local smoothing. These are the bandwidth parameters to be chosen from.
thresh	Threshold values to be chosen from.
nboot	Number of bootstrap samples.

Details

A jump-preserving local linear kernel smoothing is applied to estimate the discontinuous regression surface; Bootstrap samples are obtained by drawing with replacement from the residuals and the d_{KQ} is computed for the detected edges of the original sample and those of the bootstrap samples.

Value

Returns a list of the selected bandwdith, the selected threshold value, and a matrix of d_{KQ} values with each entry corresponding to each combination of bandwdith and threshold.

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

See Also

stepEdgeParSelLCK, stepEdgeParSelLLK, stepEdgeParSelLC2K, stepEdgeLL2K

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
#parSel = stepEdgeParSelLL2K(image=sar, bandwidth=5:6, thresh=
#20:21, nboot = 10) # Time consuming
```

`stepEdgeParSelLLK` *edge detection, parameter selection*

Description

Select threshold value for LLK edge detector using bootstrap procedure

Usage

```
stepEdgeParSelLLK(image, bandwidth, thresh, nboot)
```

Arguments

`image`	A square matrix object of size n by n, no missing value allowed.
`bandwidth`	Positive integers to specify the number of pixels used in the local smoothing. These are the bandwidth parameters to be chosen from.
`thresh`	Threshold values to be chosen from.
`nboot`	Number of bootstrap samples.

Details

A jump-preserving local linear kernel smoothing is applied to estimate the discontinuous regression surface; Bootstrap samples are obtained by drawing with replacement from the residuals and the d_{KQ} is computed for the detected edges of the original sample and those of the bootstrap samples.

Value

Returns a list of the selected bandwdith, the selected threshold value, and a matrix of d_{KQ} values with each entry corresponding to each combination of bandwdith and threshold.

References

Kang, Y., and Qiu, P., "Jump Detection in Blurred Regression Surfaces," *Technometrics*, **56**, 2014, 539-550.

See Also

`stepEdgeParSelLCK`, `stepEdgeParSelLC2K`, `stepEdgeParSelLL2K`, `stepEdgeLLK`

Examples

```
data(sar) # SAR image is bundled with the package and it is a
          # standard test image in statistics literature.
#parSel = stepEdgeParSelLLK(image=sar, bandwidth=c(5,9), thresh=
#c(17,21), nboot=10) # Time consuming
```

stopsign *Image of Stop Sign*

Description

This file contains data of stop sign image. The image has 160x160 pixels. Gray levels are in the range [0, 255]. In the data file, observations are listed as a 160x160 matrix. This image has much blurring involved.

Usage

```
stopsign
```

Format

This dataset is saved in an ascii file.

surfaceCluster *Denoising, deblurring and edge-preserving*

Description

Estimate surface using local pixel clustering and kernel smoothing. Bandwidth is specified by user.

Usage

```
surfaceCluster(image, bandwidth, sig.level, sigma, phi0, mean_std_abs, cw=3,
blur = FALSE, plot = FALSE)
```

Arguments

image	A square matrix object of size n by n, no missing value allowed.
bandwidth	A positive integer that specifies the number of pixels used in the local smoothing.
sig.level	Specifies the significance level of the hypothesis test deciding to cluster pixels or not.
sigma	Specifies the noise level (i.e., standard deviation of the error distribution). It is used for computing the asymptotic threshold for residuals, which are defined to be the difference between the local linear kernel smoothing output and the center weighted median filter output. If not specified by the user, a jump-preserving local linear kernel smoothing surface estimation (Qiu 2009) is used to obtain an estimated sigma.

phi0 Specifies the density of the standardized error distribution at 0. It is used for computing the asymptotic threshold for residuals, which are defined to be the difference between the local linear kernel smoothing output and the center weighted median filter output. If not specified by the user, a jump-preserving local linear kernel smoothing surface estimation (Qiu 2009) is used to obtain an estimated value.

mean_std_abs Specifies the mean of absolute value of the standardized error. It is used for computing the asymptotic threshold for residuals, which are defined to be the difference between the local linear kernel smoothing output and the center weighted median filter output. If not specified by the user, a jump-preserving local linear kernel smoothing surface estimation (Qiu 2009) is used to obtain an estimated value.

cw Specifies the center weight for the center weighted median filter. It must be a positive integer.

blur If blur = TRUE, besides a conventional 2-D kernel function, a univariate increasing kernel function is used in the local kernel smoothing to address the issue with blur.

plot If plot = TRUE, the image of the fitted surface is plotted

Value

Returns a list. 'estImg' is the restored image. 'sigma' is the estimated standard deviation of the random error. It is the input value if specified by the user. 'phi0' is the estimated density of the error distribution at 0. It is the input value if specified by the user. 'mean_std_abs' is the estimated absolute mean of the error distribution. It is the input value if specified by the user.

References

Kang, Y., Mukherjee, P.S., and Qiu, P. (2017), "Efficient Blind Image Deblurring Using Nonparametric Regression and Local Pixel Clustering", Technometrics, DOI: 10.1080/00401706.2017.1415975.

See Also

JPLLK_surface, threeStage

Examples

```
data(brain)
fit = surfaceCluster(image=brain, bandwidth=4, sig.level=.9995, cw=3, blur=TRUE)
```

---

surfaceCluster_bandwidth

Denoising, deblurring, bandwidth selection, and edge-preserving

Description

Select the bandwidth parameter for the function surfaceCluster based on cross validation. In the cases when there is no blur involved (i.e., denoising only), leave-one-out cross validation is used. In the cases when there is blur involved, a modified cross validation is used.

Usage

```
surfaceCluster_bandwidth(image, bandwidths, sig.level, sigma,
 phi0, mean_std_abs, relwt=0.5, cw=3, blur=FALSE)
```

Arguments

`image`	A square matrix object of size n by n, no missing value allowed.
`bandwidths`	An array of positive integers that specifies the candiate bandwidth parameters. All the array elements must be positive integers because the bandwidth is specified in terms of number of pixels.
`sig.level`	Specifies the significance level of the hypothesis test deciding to cluster pixels or not.
`sigma`	Specifies the noise level (i.e., standard deviation of the error distribution). It is used for computing the asymptotic threshold for residuals, which are defined to be the difference between the local linear kernel smoothing output and the center weighted median filter output. If not specified by the user, a jump-preserving local linear kernel smoothing surface estimation (Qiu 2009) is used to obtain an estimated sigma.
`phi0`	Specifies the density of the standardized error distribution at 0. It is used for computing the asymptotic threshold for residuals, which are defined to be the difference between the local linear kernel smoothing output and the center weighted median filter output. If not specified by the user, a jump-preserving local linear kernel smoothing surface estimation (Qiu 2009) is used to obtain an estimated value.
`mean_std_abs`	Specifies the mean of absolute value of the standardized error. It is used for computing the asymptotic threshold for residuals, which are defined to be the difference between the local linear kernel smoothing output and the center weighted median filter output. If not specified by the user, a jump-preserving local linear kernel smoothing surface estimation (Qiu 2009) is used to obtain an estimated value.
`relwt`	The relative weight assigned to the cross validation score in the continuity region. That is, 1 - relwt is assigned to the cross validation score around the step edges. It is used only when there is blur involved.
`cw`	Specifies the center weight for the center weighted median filter. It must be a positive integer.
`blur`	If blur = TRUE, besides a conventional 2-D kernel function, a univariate increasing kernel function is used in the local kernel smoothing to address the issue with blur.

Value

Returns a list. 'cv_dataframe' contains the cross validation scores corresponding to each candidate bandwidth. 'bandwidth_hat' is the selected bandwidth. 'sigma' is the estimated standard deviation of the random error. It is the input value if specified by the user. 'phi0' is the estimated density of the error distribution at 0. It is the input value if specified by the user. 'mean_std_abs' is the estimated absolute mean of the error distribution. It is the input value if specified by the user.

References

Kang, Y., Mukherjee, P.S., and Qiu, P. (2017), "Efficient Blind Image Deblurring Using Nonparametric Regression and Local Pixel Clustering", Technometrics, DOI: 10.1080/00401706.2017.1415975.

Qiu, P., "Jump-preserving surface reconstruction from noisy data," Annals of the Institute of Statistical Mathematics, 61(3), 2009, 715–751.

See Also

`JPLLK_surface`, `threeStage`

Examples

```
data(brain)
bandwidth_select = surfaceCluster_bandwidth(image=brain,
bandwidths=c(3:4), sig.level=.9995, blur=TRUE)
```

---

threeStage *Denoising, deblurring and edge-preserving*

Description

Estimate surface using local smoothing and fitting principal component line. Bandwidth is specified by user.

Usage

```
threeStage(image, bandwidth, edge1, edge2,
blur = FALSE, plot = FALSE)
```

Arguments

`image`	A square matrix object of size n by n, no missing value allowed.
`bandwidth`	A positive integer that specifies the number of pixels used in the local smoothing.
`edge1`	A matrix of 0 and 1 of the same size as image represents detected step edge pixels
`edge2`	A matrix of 0 and 1 of the same size as image represents detected roof/valley edge pixels

`blur`	If blur = TRUE, besides a conventional 2-D kernel function, a univariate increasing kernel function is used in the local kernel smoothing to address the issue with blur.
`plot`	If plot = TRUE, the image of the fitted surface is plotted

Details

At each pixel, if there are step edges detected in the local neighborhood, a principal component line is fitted through the detected edge pixels to approximate the step edge locally and then the regression surface is estimated by a local constant kernel smoothing procedure using only the pixels on one side of the principal component line. If there are no step edges but roof/valley edges detected in the local neighborhood, the same procedure is followed except that the principal component line to fitted through the detected roof/valley edge pixels. In cases when there is either no step edges or roof/valley edges detected in the neighborhood, the regression surface at the pixel is estimated by the conventional local linear kernel smoothing procedure.

Value

Returns the restored image, which is represented by a matrix

References

Qiu, P., and Kang, Y. "Blind Image Deblurring Using Jump Regression Analysis," *Statistica Sinica*, **25**, 2015, 879-899.

See Also

`JPLLK_surface`, `surfaceCluster`

Examples

```
data(sar)
stepEdge = stepEdgeLCK(sar, bandwidth=4, thresh=20)
stepEdge1 = modify2(bandwidth=4, stepEdge)
fit = threeStage(image=sar, bandwidth=4, edge1=
stepEdge1, edge2=array(0, rep(ncol(sar), 2)))
```

`threeStageParSel` *image denoising/deblurring, bandwidth selection, bootstrap*

Description

Select the bandwidth value for the image restoration method implemented in the function threeStage

Usage

```
threeStageParSel(image, bandwidth, edge1, edge2, nboot, blur=FALSE)
```

Arguments

`image`	A square matrix object of size n by n, no missing value allowed.
`bandwidth`	Bandwidth values to be chosen from. Each of these values need to be an positive integer which specifies the number of pixels used in the local smoothing.
`edge1`	A matrix of 0 and 1 of the same size as image represents detected step edge pixels.
`edge2`	A matrix of 0 and 1 of the same size as image represents detected roof/valley edge pixels.
`nboot`	Required when blur is TRUE. Unused when blur is FALSE. An positive integer to specify the number of bootstraps to perform. See Qiu and Kang (2015) Statistica Sinica for details.
`blur`	TRUE if the image contains blur, FALSE otherwise. If TRUE, the hybrid selection method proposed in Qiu and Kang (2015) Statistica Sinica is used. If FALSE, the leave-one-out cross validation is used.

Value

Returns a list of the selected bandwdith, and a matrix of CV values with each entry corresponding to each choice of bandwdith.

References

Qiu, P., and Kang, Y. "Blind Image Deblurring Using Jump Regression Analysis," *Statistica Sinica*, **25**, 2015, 879-899.

Examples

```
data(peppers) # Peppers image is bundled with the package and it is a
          # standard test image in image processing literature.
# Not run
#step.edges = stepEdgeLLK(peppers, 9, 17) # Step edge detection
#roof.edges = roofEdge(peppers, 6, 3000, edge1=step.edges) # Roof edge detection
#parSel = threeStageParSel(image=peppers, edge1=step.edges,
#edge2=roof.edges, bandwidth=4:5) # Time consuming
```

Index