

Package ‘image.ContourDetector’

November 27, 2025

Type Package
Title Implementation of the Unsupervised Smooth Contour Line Detection for Images
Description An implementation of the Unsupervised Smooth Contour Detection algorithm for digital images as described in the paper: ``Unsupervised Smooth Contour Detection" by Rafael Grompone von Gioi, and Gregory Randall (2016). The algorithm is explained at <doi:10.5201/ipol.2016.175>.
Maintainer Jan Wijffels <jwijffels@bnosac.be>
Encoding UTF-8
License AGPL-3
Version 0.1.2
URL <https://github.com/bnosac/image>
Imports Rcpp (>= 0.12.8), sp
LinkingTo Rcpp
Suggests pixmap, magick, raster
RoxygenNote 7.3.2
NeedsCompilation yes
Author Jan Wijffels [aut, cre, cph] (R wrapper),
BNOSAC [cph] (R wrapper),
Rafael Grompone von Gioi [ctb, cph] (src/smooth_contours),
Gregory Randall [ctb, cph] (src/smooth_contours),
Niccolò Marchi [ctb]
Repository CRAN
Date/Publication 2025-11-27 20:50:07 UTC

Contents

image.ContourDetector-package	2
image_contour_detector	3
plot.cld	4
Index	6

image.ContourDetector-package

The image.ContourDetector package detects contour lines in images

Description

Unsupervised Smooth Contour Detection.

Following the a contrario approach, the starting point is defining the conditions where contours should not be detected: soft gradient regions contaminated by noise. To achieve this, low frequencies are removed from the input image. Then, contours are validated as the frontiers separating two adjacent regions, one with significantly larger values than the other. Significance is evaluated using the Mann-Whitney U test to determine whether the samples were drawn from the same distribution or not. This test makes no assumption on the distributions. The resulting algorithm is similar to the classic Marr-Hildreth edge detector, with the addition of the statistical validation step. Combined with heuristics based on the Canny and Devernay methods, an efficient algorithm is derived producing sub-pixel contours.

Author(s)

Maintainer: Jan Wijffels <jwi.jffels@bnosac.be> (R wrapper) [copyright holder]

Other contributors:

- BNOSAC (R wrapper) [copyright holder]
- Rafael Grompone von Gioi <grompone@gmail.com> (src/smooth_contours) [contributor, copyright holder]
- Gregory Randall <randall@fing.edu.uy> (src/smooth_contours) [contributor, copyright holder]
- Niccolò Marchi <niccolo.marchi@unipd.it> [contributor]

References

Rafael Grompone von Gioi, and Gregory Randall, Unsupervised Smooth Contour Detection, Image Processing On Line, 6 (2016), pp. 233-267. [doi:10.5201/ipol.2016.175](https://doi.org/10.5201/ipol.2016.175)

See Also

[image_contour_detector](#)

image_contour_detector*Unsupervised Smooth Contour Lines Detection in an image*

Description

Unsupervised Smooth Contour Detection.

Following the a contrario approach, the starting point is defining the conditions where contours should not be detected: soft gradient regions contaminated by noise. To achieve this, low frequencies are removed from the input image. Then, contours are validated as the frontiers separating two adjacent regions, one with significantly larger values than the other. Significance is evaluated using the Mann-Whitney U test to determine whether the samples were drawn from the same distribution or not. This test makes no assumption on the distributions. The resulting algorithm is similar to the classic Marr-Hildreth edge detector, with the addition of the statistical validation step. Combined with heuristics based on the Canny and Devernay methods, an efficient algorithm is derived producing sub-pixel contours.

Usage

```
image_contour_detector(x, Q = 2, ...)
```

Arguments

x	a matrix of image pixel values in the 0-255 range.
Q	numeric value with the pixel quantization step
...	further arguments, not used yet

Value

an object of class `cld` which is a list with the following elements

- `curves`: The number of contour lines found
- `contourpoints`: The number of points defining the contour lines found
- `data`: A `data.frame` with columns 'x', 'y' and 'curve' giving the x/y locations for each contour curve

References

Rafael Grompone von Gioi, and Gregory Randall, Unsupervised Smooth Contour Detection, Image Processing On Line, 6 (2016), pp. 233-267. [doi:10.5201/ipol.2016.175](https://doi.org/10.5201/ipol.2016.175)

Examples

```
library(pixmap)
imagelocation <- system.file("extdata", "image.pgm", package="image.ContourDetector")
image          <- read.pnm(file = imagelocation, cellres = 1)

x              <- image@grey * 255
contourlines   <- image_contour_detector(x, Q = 2)
contourlines
plot(image)
plot(contourlines, add = TRUE, col = "red")

##
## line_segment_detector expects a matrix as input
## if you have a jpg/png/... convert it to pgm first or take the r/g/b channel

library(magick)
x <- image_read(system.file("extdata", "atomium.jpg", package="image.ContourDetector"))
x
mat <- image_data(x, channels = "gray")
mat <- as.integer(mat, transpose = TRUE)
mat <- drop(mat)
contourlines <- image_contour_detector(mat)
plot(contourlines)

##
## working with a RasterLayer
##

library(raster)
x <- raster(system.file("extdata", "landscape.tif", package="image.ContourDetector"))

contourlines <- image_contour_detector(x)
image(x)
plot(contourlines, add = TRUE, col = "blue", lwd = 10)
```

plot.cld

Plot the detected contour lines from the image_contour_detector

Description

Plot the detected contour lines from the image_contour_detector

Usage

```
## S3 method for class 'cld'
plot(x, ...)
```

Arguments

`x` an object of class `cld` as returned by [image_contour_detector](#)
`...` further arguments passed on to `plot`

Value

invisibly a `SpatialLines` object with the contour lines

Examples

```
library(pixmap)
imagelocation <- system.file("extdata", "image.pgm", package="image.ContourDetector")
image <- read.pnm(file = imagelocation, cellres = 1)
contourlines <- image_contour_detector(image@grey * 255)
plot(image)
plot(contourlines, add = TRUE, col = "red")
```

Index

`image.ContourDetector`
 (`image.ContourDetector-package`),
 [2](#)
`image.ContourDetector-package`, [2](#)
`image_contour_detector`, [2](#), [3](#), [5](#)
`plot.cld`, [4](#)