

# Package ‘image.CannyEdges’

August 1, 2020

**Type** Package

**Title** Implementation of the Canny Edge Detector for Images

**Version** 0.1.0

**Maintainer** Jan Wijffels <jwijffels@bnosac.be>

**Description** An implementation of the Canny Edge Detector for detecting edges in images. The package provides an interface to the algorithm available at <<https://github.com/Neseb/canny>>.

**License** GPL-3

**URL** <https://github.com/bnosac/image>

**Encoding** UTF-8

**Imports** Rcpp (>= 0.12.9)

**LinkingTo** Rcpp

**Suggests** pixmap, magick

**RoxygenNote** 7.1.0

**SystemRequirements** libpng, fftw3

**NeedsCompilation** yes

**Author** Jan Wijffels [aut, cre, cph],  
BNOSAC [cph],  
Vincent Maioli [ctb, cph],  
IPOL Image Processing On Line [cph]

**Repository** CRAN

**Date/Publication** 2020-08-01 11:00:02 UTC

## R topics documented:

image.CannyEdges-package	2
image_canny_edge_detector	2
plot.image_canny	3

<b>Index</b>	<b>5</b>
--------------	----------

---

image.CannyEdges-package

*Implementation of the Canny Edge Detector for Images*

---

### Description

Canny Edge Detector for Images. See [https://en.wikipedia.org/wiki/Canny\\_edge\\_detector](https://en.wikipedia.org/wiki/Canny_edge_detector). Adapted from <https://github.com/Neseb/canny>.

### See Also

[image\\_canny\\_edge\\_detector](#)

---

image\_canny\_edge\_detector

*Canny Edge Detector for Images*

---

### Description

Canny Edge Detector for Images. See [https://en.wikipedia.org/wiki/Canny\\_edge\\_detector](https://en.wikipedia.org/wiki/Canny_edge_detector). Adapted from <https://github.com/Neseb/canny>.

### Usage

```
image_canny_edge_detector(x, s = 2, low_thr = 3, high_thr = 10, accGrad = TRUE)
```

### Arguments

x	a matrix of image pixel values in the 0-255 range.
s	sigma, the Gaussian filter variance. Defaults to 2.
low_thr	lower threshold value of the algorithm. Defaults to 3.
high_thr	upper threshold value of the algorithm. Defaults to 10
accGrad	logical indicating to trigger higher-order gradient

### Value

a list with element edges which is a matrix with values 0 or 255 indicating in the same dimension of x. Next to that the list also contains the input parameters s, low\_thr, high\_thr and accGrad, the number of rows (nx) and columns of the image (ny) and the number of pixels which have value 255 (pixels\_nonzero).

## Examples

```
if(requireNamespace("pixmap") && requireNamespace("magick")){

  library(pixmap)
  imagelocation <- system.file("extdata", "chairs.pgm", package="image.CannyEdges")
  image <- read.pnm(file = imagelocation, cellres = 1)
  x <- image@grey * 255

  edges <- image_canny_edge_detector(x)
  edges
  plot(edges)

  ##
  ## image_canny_edge_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.CannyEdges"))
  x
  image <- image_data(x, channels = "Gray")
  image <- as.integer(image, transpose = TRUE)
  edges <- image_canny_edge_detector(image)
  plot(edges)

  f <- tempfile(fileext = ".pgm")
  library(magick)
  x <- image_read(system.file("extdata", "atomium.jpg", package="image.CannyEdges"))
  x <- image_convert(x, format = "pgm", depth = 8)
  image_write(x, path = f, format = "pgm")

  image <- read.pnm(f, cellres = 1)
  edges <- image_canny_edge_detector(image@grey * 255)
  plot(edges)

  file.remove(f)
}
```

---

plot.image\_canny

*Plot the result of the Canny Edge Detector*

---

## Description

Plot the result of [image\\_canny\\_edge\\_detector](#)

## Usage

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

**Arguments**

`x` an object of class `image_canny` as returned by `image_canny_edge_detector`  
`...` further arguments passed on to `plot`, except `type`, `xlab` and `ylab` which are set inside the function

**Value**

`invisible()`

**Examples**

```
library(pixmap)
imagelocation <- system.file("extdata", "chairs.pgm", package="image.CannyEdges")
image <- read.pnm(file = imagelocation, cellres = 1)
edges <- image_canny_edge_detector(image@grey * 255)
plot(edges)
```

# Index

`image.CannyEdges`-package, [2](#)  
`image_canny_edge_detector`, [2](#), [2](#), [3](#), [4](#)  
`plot.image_canny`, [3](#)