Package 'image.dlib'

October 18, 2022

Type Package

Title Image Processing Functionality using the 'dlib' Package

Version 0.1.1

Maintainer Jan Wijffels <jwijffels@bnosac.be>

Description Facility wrappers around the image processing functionality of 'dlib'. 'Dlib' <http://dlib.net> is a 'C++' toolkit containing machine learning algorithms and computer vision tools. Currently the package allows to find feature descriptors of digital images, in particular 'SURF' and 'HOG' descriptors.

License BSL-1.0

URL https://github.com/bnosac/image

Imports Rcpp (>= 0.12.9)

LinkingTo Rcpp

Suggests magick, FNN

SystemRequirements C++17

RoxygenNote 7.1.2

NeedsCompilation yes

Author Jan Wijffels [aut, cre, cph] (R wrapper),
BNOSAC [cph] (R wrapper),
Davis E. King [ctb, cph] (Main contributor of dlib - files in inst/dlib),
dlib authors [ctb, cph] (see file AUTHORS)

Repository CRAN

Date/Publication 2022-10-18 08:02:38 UTC

R topics documented:

image_fhog 2
image_surf 3

Index **6**

image_fhog *HOG (Histogram Of Gradients) feature extraction*

Description

Gets the histogram of oriented gradients (HOG) feature descriptor https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients of an image. More information on the calculation, see the reference paper or the dlib reference http://dlib.net/dlib/image_transforms/fhog_abstract.h.html

Usage

```
image_fhog(
  x,
  cell_size = 8L,
  filter_rows_padding = 1L,
  filter_cols_padding = 1L
)
```

Arguments

x	a 3-dimensional array of integer pixel values where the first dimension is RGB, 2nd the width of the image and the 3rd the height of the image. See the example.
cell_size	integer. HOG groups pixels in cells, based on which a histogram of the gradiens is computed. Give the cell size. Defaults to 8.
filter_rows_padding	integer indicating how many extra rows and columns of zero padding along the borders to add, for more efficient convolution code
filter_cols_padding	integer indicating how many extra rows and columns of zero padding along the borders to add, for more efficient convolution code

Value

a list with HOG features of the pixels mapped into HOG space. It contains:

- hog_height: The size of the height of the image in HOG space
- hog_width: The size of the width of the image in HOG space
- hog_feature: The 31 dimensional FHOG vector, one for each pixel in HOG space. This is an array of dimension hog_height x hog_width x 31. The 31-dimensional vector describes the gradient structure within the cell.
- hog_cell_size: The HOG cell_size
- filter_rows_padding: The value of filter_rows_padding
- filter_cols_padding: The value of filter_cols_padding

References

Object Detection with Discriminatively Trained Part Based Models by P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9, Sep. 2010

Examples

```
library(magick)
f <- system.file("extdata", "cruise_boat.png", package = "image.dlib")
x <- image_read(f)
x
feats <- image_data(x, channels = "rgb")
feats <- as.integer(feats, transpose = FALSE)
feats <- image_fhog(feats, cell_size = 8)
str(feats)

## FHOG feature vector of pixel at HOG space position 1, 1
feats$fhog[1, 1, ]
## FHOG feature vector of pixel at HOG space position 4 (height), 7 (width)
feats$fhog[4, 7, ]
```

---

image_surf *Find SURF points in an image*

Description

SURF (speeded up robust features) points are blobs in a digital image https://en.wikipedia.org/wiki/Speeded_up_robust_features. The function identifies these points in the image and also provides a numeric descriptor of each point.

This SURF descriptor is used on computer vision applications like object matching (by comparing the SURF descriptor of several images using e.g. kNN - from the rflann package) or object recognition. The SURF feature descriptor is based on the sum of the Haar wavelet response around the point of interest. More information on the calculation, see the reference paper.

Usage

```
image_surf(x, max_points = 1000, detection_threshold = 30)
```

Arguments

x	a 3-dimensional array of integer pixel values where the first dimension is RGB, 2nd the width of the image and the 3rd the height of the image. See the example.
max_points	maximum number of surf points to find
detection_threshold	detection threshold (the higher, the lower the number of SURF points found)

Value

a list with SURF points found and the SURF descriptor. It contains:

- points: The number of SURF points
- x: The x location of each SURF point
- y: The y location of each SURF point
- angle: The angle of each SURF point
- pyramid_scale: The pyramid scale of each SURF point
- score: The score of each SURF point
- laplacian: The laplacian at each SURF point
- surf: The SURF descriptor which is a matrix with 64 columns describing the point

References

SURF: Speeded Up Robust Features By Herbert Bay, Tinne Tuytelaars, and Luc Van Gool

Examples

```
library(magick)
f <- system.file("extdata", "cruise_boat.png", package = "image.dlib")
x <- image_read(f)
x
surf_blobs <- image_data(x, channels = "rgb")
surf_blobs <- as.integer(surf_blobs, transpose = FALSE)
surf_blobs <- image_surf(surf_blobs, max_points = 10000, detection_threshold = 50)
str(surf_blobs)

library(magick)
plt <- image_draw(x)
points(surf_blobs$x, surf_blobs$y, col = "red", pch = 20)
dev.off()
plt
## Not run:
## Plot the points
"as.cimg.magick-image" <- function(x){
  out <- lapply(x, FUN=function(frame){
    frame <- as.integer(frame[[1]])[, , 1:3] # dropping the alpha channel
    dim(frame) <- append(dim(frame), 1, after = 2)
    frame
  })
  out$along <- 3
  out <- do.call(abind::abind, out)
  out <- imager::as.cimg(out)
  out <- imager::permute_axes(out, "yxzc")
  out
}
library(imager)
library(magick)
library(abind)
```

```
img <- image_read(path = f)
plot(as.cimg(img), main = "SURF points")
points(surf_blobs$x, surf_blobs$y, col = "red", pch = 20)

## End(Not run)

library(magick)
img1   <- image_scale(logo, "x300")
img2   <- image_flip(img1)
height <- image_info(img1)$height
sp1 <- image_surf(image_data(img1, channels = "rgb"), max_points = 50)
sp2 <- image_surf(image_data(img2, channels = "rgb"), max_points = 50)

## Match surf points and plot matches
library(FNN)
k <- get.knnx(sp1$surf, sp2$surf, k = 1)
combined <- image_append(c(img1, img2), stack = TRUE)
plt <- image_draw(combined)
points(sp1$x, sp1$y, col = "red")
points(sp2$x, sp2$y + height, col = "blue")
for(i_from in head(order(k$nn.dist), 50)){
  i_to   <- k$nn.index[i_from]
 lines(x = c(sp1$x[i_to], sp2$x[i_from]), y = c(sp1$y[i_to], sp2$y[i_from] + height), col = "red")
}
dev.off()
plt
```

Index

image_fhog, 2
image_surf, 3