Package ‘fastGLCM’

October 13, 2022

Type Package

Title 'GLCM' Texture Features

Version 1.0.2

Date 2022-09-25

Maintainer Lampros Mouselimis <mouselimislampros@gmail.com>

BugReports https://github.com/mlampros/fastGLCM/issues

URL https://github.com/mlampros/fastGLCM

Description Two 'Gray Level Co-occurrence Matrix' ('GLCM') implementations are included: The first is a fast 'GLCM' feature texture computation based on 'Python' 'Numpy' arrays ('Github' Repository, <https://github.com/tzm030329/GLCM>). The second is a fast 'GLCM' 'RcppArmadillo' implementation which is parallelized (using 'OpenMP') with the option to return all 'GLCM' features at once. For more information, see ``Artifact-Free Thin Cloud Removal Using Gans'' by Toizumi Takahiro, Zini Simone, Sagi Kazutoshi, Kaneko Eiji, Tsukada Masato, Schettini Raimondo (2019), IEEE International Conference on Image Processing (ICIP), pp. 3596-3600, <doi:10.1109/ICIP.2019.8803652>.

SystemRequirements apt-get-pip: apt-get install -y python3-pip (deb), python3-pip: python3 -m pip install -U pip (deb), numpy: pip3 install -U numpy (deb), cv2: pip3 install -U opencv-python (deb), matplotlib: pip3 install -U matplotlib (deb), skimage: pip3 install -U scikit-image (deb), libarmadillo: apt-get install -y libarmadillo-dev (deb), libblas: apt-get install -y libblas-dev (deb), liblapack: apt-get install -y liblapack-dev (deb), libarpack++2: apt-get install -y libarpack++2-dev (deb), gfortran: apt-get install -y gfortran (deb)

License GPL-3

Copyright inst/COPYRIGHTS

Depends R(>= 3.2.3)

Imports Rcpp (>= 1.0.8.3), R6, rlang, OpenImageR, utils

LinkingTo Rcpp, RcppArmadillo, OpenImageR

Suggests reticulate, covr, knitr, rmarkdown, testthat (>= 3.0.0)

Encoding UTF-8

RoxygenNote 7.2.1

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

Author Lampros Mouselimis [aut, cre] (<https://orcid.org/0000-0002-8024-1546>),
Takahiro Toizumi [cph] (Author of the fastGLCM Python code)

Repository CRAN

Date/Publication 2022-09-25 17:50:09 UTC

R topics documented:

`fastglcm` *GLCM feature texture extraction*

Description

GLCM feature texture extraction

GLCM feature texture extraction

Usage

```
# init <- fastglcm$new()
```

Methods

```
fastglcm$new()

--------------

GLCM_compute()

--------------
```

Methods

Public methods:

- `fastglcm$new()`
- `fastglcm$GLCM_compute()`
- `fastglcm$clone()`

Method `new()`**:** Initialization method for the 'fastglcm' R6 class

Usage:

```
fastglcm$new()
```

Method `GLCM_compute()`**:** The GLCM computation method to receive the results

Usage:

```
fastglcm$GLCM_compute(
  img,
  method,
  vmin = 0,
  vmax = 255,
  levels = 8,
  ks = 5,
  distance = 1,
  angle = 0,
  verbose = FALSE
)
```

Arguments:

`img` a numeric matrix

`method` a character string specifying the method. Can be one of 'mean', 'std', 'contrast', 'dissimilarity', 'homogeneity', 'ASM_Energy', 'max' or 'entropy'

`vmin` a numeric value specifying the minimum value of the input image (*img*)

`vmax` a numeric value specifying the maximum value of the input image (*img*)

`levels` an integer specifying the window size. This parameter will create a mask of size *levels x levels* internally

`ks` an integer specifying the kernel size. A kernel of 1's will be created and the *cv2.filter2D* filter will be utilized for the convolution

`distance` a numeric value specifying the pixel pair distance offsets (a 'pixel' value such as 1.0, 2.0 etc.)

`angle` a numeric value specifying the pixel pair angles (a 'degree' value such as 0.0, 30.0, 45.0, 90.0 etc.)

`verbose` a boolean. If TRUE then information will be printed out in the console

Returns: a list object if the method is set to 'ASM_Energy' otherwise a numeric matrix

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
fastglcm$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

https://github.com/tzm030329/GLCM https://github.com/1044197988/Python-Image-feature-extraction

Examples

```
## Not run:

require(fastGLCM)
require(OpenImageR)

file_im = system.file('images', 'Sugar_Cane_Bolivia_PlanetNICFI.png', package = 'fastGLCM')
im = readImage(file_im)

#..................................
# convert to gray and make sure that
# pixel values are between 0 and 255
#..................................

im = rgb_2gray(im)
im = im * 255

MIN = min(as.vector(im))
MAX = max(as.vector(im))

#...............
# methods to use
#...............

methods_py = c('mean',
               'std',
               'contrast',
               'dissimilarity',
               'homogeneity',
               'ASM_Energy',
               'max',
               'entropy')

init = fastglcm$new()

lst_glcm_py = list()

for (item_m in methods_py) {

  cat(paste0('Method: ', item_m), '\n')

  res_item = init$GLCM_compute(img = im,
                               method = item_m,
                               vmin = as.integer(MIN),
                               vmax = as.integer(MAX),
                               levels = as.integer(8),
                               ks = as.integer(5),
```

```
                                distance = 1.0,
                                angle = 0.0)

  lst_glcm_py[[item_m]] = res_item
}

#..............................
# Create two different sublists
# for 'ASM' and 'Energy'
#..............................

lst_glcm_py = append(lst_glcm_py, list(lst_glcm_py[['ASM_Energy']][[1]]), after = 5)
names(lst_glcm_py)[6] = 'ASM'

lst_glcm_py = append(lst_glcm_py, list(lst_glcm_py[['ASM_Energy']][[2]]), after = 6)
names(lst_glcm_py)[7] = 'energy'

lst_glcm_py[['ASM_Energy']] = NULL

str(lst_glcm_py)

#.........................
# multi-plot of the output
#.........................

plot_multi_images(list_images = lst_glcm_py,
                  par_ROWS = 2,
                  par_COLS = 5,
                  titles = names(lst_glcm_py))

## End(Not run)
```

fastGLCM_Rcpp *GLCM feature texture extraction*

Description

GLCM feature texture extraction

Usage

```
fastGLCM_Rcpp(
  data,
  methods,
  levels = 8,
  kernel_size = 5,
  distance = 1,
  angle = 0,
  dir_save = NULL,
```

```
  threads = 1,
  verbose = FALSE
)
```

Arguments

`data`	a numeric matrix
`methods`	a vector of character strings. One or all of the following: 'mean', 'std', 'contrast', 'dissimilarity', 'homogeneity', 'ASM', 'energy', 'max', 'entropy'
`levels`	an integer specifying the window size. This parameter will create a mask of size *levels x levels* internally
`kernel_size`	an integer specifying the kernel size. A kernel of 1's will be created and the *cv2.filter2D* filter will be utilized for the convolution
`distance`	a numeric value specifying the pixel pair distance offsets (a 'pixel' value such as 1.0, 2.0 etc.)
`angle`	a numeric value specifying the pixel pair angles (a 'degree' value such as 0.0, 30.0, 45.0, 90.0 etc.)
`dir_save`	either NULL or a character string specifying a valid path to a directory where the output GLCM matrices (for the specified 'methods') will be saved. By setting this parameter to a valid directory the memory usage will be decreased.
`threads`	an integer value specifying the number of cores to run in parallel
`verbose`	a boolean. If TRUE then information will be printed out in the console

Details

The following are two factors which (highly probable) will increase memory usage during computations:

- **1st.** the image size (the user might have to resize the image first)
- **2nd.** the 'levels' parameter. The bigger this parameter the more matrices will be initialized and more memory will be used. For instance if the 'levels' parameter equals to 8 then 8 * 8 = 64 matrices of equal size to the input image will be initialized. That means if the image has dimensions (2745 x 2745) and the image-object size is approx. 60 MB then by initializing 64 matrices the memory will increase to 3.86 GB.

This function is an Rcpp implementation of the python fastGLCM module. When using each function separately by utilizing all threads it's slightly faster compared to the python vectorized functions, however it's a lot faster when computing all features at once.

The **dir_save** parameter allows the user to save the GLCM's as .csv files to a directory. That way the output GLCM's matrices won't be returned in the R session (reduced memory usage). However, by saving the GLCM's to .csv files the computation time increases.

Value

a list consisting of one or more GLCM features

References

https://github.com/tzm030329/GLCM

Examples

```
require(fastGLCM)
require(OpenImageR)
require(utils)

temp_dir = tempdir(check = FALSE)
# temp_dir

zip_file = system.file('images', 'JAXA_Joso-City2_PAN.tif.zip', package = "fastGLCM")
utils::unzip(zip_file, exdir = temp_dir)
path_extracted = file.path(temp_dir, 'JAXA_Joso-City2_PAN.tif')

im = readImage(path = path_extracted)
dim(im)

#...............................................
# resize the image and adjust pixel values range
#...............................................

im = resizeImage(im, 500, 500, 'nearest')
im = OpenImageR::norm_matrix_range(im, 0, 255)

#---------------------------------
# computation of all GLCM features
#---------------------------------

methods = c('mean',
            'std',
            'contrast',
            'dissimilarity',
            'homogeneity',
            'ASM',
            'energy',
            'max',
            'entropy')

res_glcm = fastGLCM_Rcpp(data = im,
                         methods = methods,
                         levels = 8,
                         kernel_size = 5,
                         distance = 1.0,
                         angle = 0.0,
                         threads = 1)
# str(res_glcm)

# plot_multi_images(list_images = res_glcm,
#                   par_ROWS = 2,
```

```
#                   par_COLS = 5,
#                   titles = methods)

if (file.exists(path_extracted)) file.remove(path_extracted)
```

`plot_multi_images` *Plot multiple images*

Description

Plot multiple images

Usage

```
plot_multi_images(list_images, par_ROWS, par_COLS, ...)
```

Arguments

`list_images`	a list of images that should be visualized
`par_ROWS`	an integer specifying the number of rows of the ouput plot-grid
`par_COLS`	an integer specifying the number of columns of the output plot-grid
`...`	further arguments for the 'plot_multi_images' method of the 'GaborFeatureExtract' R6 class ('OpenImageR' package)

Details

For the usage of the 'plot_multi_images()' function see the example section of the 'fastGLCM_Rcpp()' function and 'fastglcm()' R6 class

Value

it doesn't return an R object but it displays a list of input images

Index