Package 'SuperpixelImageSegmentation'

June 13, 2020

Type Package

Title Superpixel Image Segmentation

Version 1.0.2

Date 2020-06-13

Author Lampros Mouselimis <mouselimislampros@gmail.com>

Maintainer Lampros Mouselimis <mouselimislampros@gmail.com>

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

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

Description Image Segmentation using Superpixels, Affinity Propagation and Kmeans Clustering. The R code is based primarily on the article ``Image Segmentation using SLIC Superpixels and Affinity Propagation Clustering, Bao Zhou, International Journal of Science and Research (IJSR), 2013" <https://pdfs.semanticscholar.org/6533/654973054b742e725fd433265700c07b48a2.pdf>.

License GPL-3

Encoding UTF-8

LazyData true

Depends R(>= 3.2)

Imports Rcpp (>= 0.12.10), R6, OpenImageR, grDevices, lattice

LinkingTo Rcpp, RcppArmadillo (>= 0.9.1), ClusterR, OpenImageR

Suggests testthat, covr, knitr, rmarkdown

RoxygenNote 7.1.0

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-06-13 15:40:07 UTC

R topics documented:

`Image_Segmentation` *Segmentation of images based on Superpixels, Affinity propagation and Kmeans clustering*

Description

Segmentation of images based on Superpixels, Affinity propagation and Kmeans clustering

Segmentation of images based on Superpixels, Affinity propagation and Kmeans clustering

Usage

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

Details

sim_wL, *sim_wA*, *sim_wB* are the weights of the three channels. They keep balance so as to be consistent with human perception.

The quantity *colorradius* adjusts the number of clusters, and if its value is low, the number of targets would increase, which leads to more detailed segmentation results.

If the *adjust_centroids_and_return_masks* parameter is set to FALSE then the output *kmeans_image_data* will be an RGB image, otherwise it will be a black-and-white image.

colour_type parameter: RGB (Red-Green-Blue), LAB (Lightness, A-colour-dimension, B-colour-dimension) or HSV (Hue, Saturation, Value) colour.

Higher resolution images give better results.

The *affinity propagation* algorithm is used here with default parameter values.

By setting the *sim_normalize* parameter to TRUE, the affinity propagation algorithm requires less iterations to complete. However, the *colorradius* parameter does not have an effect if the similarity matrix is normalized.

————————kmeans initializers————————————-

optimal_init : this initializer adds rows of the data incrementally, while checking that they do not already exist in the centroid-matrix

quantile_init : initialization of centroids by using the cummulative distance between observations and by removing potential duplicates

kmeans++ : kmeans++ initialization. Reference : http://theory.stanford.edu/~sergei/papers/kMeansPP-soda.pdf AND http://stackoverflow.com/questions/5466323/how-exactly-does-k-means-work

random : random selection of data rows as initial centroids

Methods

```
Image_Segmentation$new()
```

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

```
spixel_segmentation()
```

```
--------------
spixel_masks_show()
--------------
spixel_clusters_show()
--------------
```

Methods

Public methods:

- Image_Segmentation$new()
- Image_Segmentation$spixel_segmentation()
- Image_Segmentation$spixel_masks_show()
- Image_Segmentation$spixel_clusters_show()
- Image_Segmentation$clone()

Method new():

Usage:

```
Image_Segmentation$new()
```

Method spixel_segmentation():

Usage:

```
Image_Segmentation$spixel_segmentation(
  input_image,
  method = "slic",
  superpixel = 200,
  kmeans_method = "",
  AP_data = FALSE,
  use_median = TRUE,
  minib_kmeans_batch = 10,
  minib_kmeans_init_fraction = 0.5,
  kmeans_num_init = 3,
  kmeans_max_iters = 100,
  kmeans_initializer = "kmeans++",
  colour_type = "RGB",
  compactness_factor = 20,
  adjust_centroids_and_return_masks = FALSE,
  return_labels_2_dimensionsional = FALSE,
  sim_normalize = FALSE,
  sim_wL = 3,
  sim_wA = 10,
  sim_wB = 10,
  sim_color_radius = 20,
  verbose = FALSE
)
```

Arguments:

`input_image` a 3-dimensional input image (the range of the pixel values should be preferably in the range 0 to 255)

`method` a character string specifying the superpixel method. It can be either "slic" or "slico"

`superpixel` a numeric value specifying the number of superpixels

`kmeans_method` a character string specifying the kmeans method. If not empty ("") then it can be either "kmeans" or "mini_batch_kmeans"

`AP_data` a boolean. If TRUE then the affinity propagation image data will be computed and returned

`use_median` a boolean. If TRUE then the median will be used rather than the mean value for the inner computations

`minib_kmeans_batch` the size of the mini batches

`minib_kmeans_init_fraction` percentage of data to use for the initialization centroids (applies if initializer is *kmeans++* or *optimal_init*). Should be a float number between 0.0 and 1.0.

`kmeans_num_init` number of times the algorithm will be run with different centroid seeds

`kmeans_max_iters` the maximum number of clustering iterations

`kmeans_initializer` the method of initialization. One of, *optimal_init*, *quantile_init*, *kmeans++* and *random*. See details for more information

`colour_type` a character string specifying the colour type. It can be one of "RGB", "LAB" or "HSV"

`compactness_factor` a numeric value specifying the compactness parameter in case that *method* is "slic"

`adjust_centroids_and_return_masks` a boolean. If TRUE and the *kmeans_method* parameter is NOT empty ("") then the centroids will be adjusted and image-masks will be returned. This will allow me to plot the masks using the *spixel_masks_show* method.

`return_labels_2_dimensionsional` a boolean. If TRUE then a matrix of labels based on the output superpixels in combination with the Affinity Propagation clusters will be returned

`sim_normalize` a boolean. If TRUE then the constructed similarity matrix will be normalised to have unit p-norm (see the armadillo documentation for more details)

`sim_wL` a numeric value specifying the weight for the *"L"* channel of the image (see the details section for more information)

`sim_wA` a numeric value specifying the weight for the *"A"* channel of the image (see the details section for more information)

`sim_wB` a numeric value specifying the weight for the *"B"* channel of the image (see the details section for more information)

`sim_color_radius` a numeric value specifying the *colorradius* (see the details section for more information)

`verbose` a boolean. If TRUE then information will be printed in the console (spixel_masks_show method)

Method `spixel_masks_show()`**:**

Usage:

```
Image_Segmentation$spixel_masks_show(
  delay_display_seconds = 3,
  display_all = FALSE,
```

```
  margin_btw_plots = 0.15,
  verbose = FALSE
)
```

Arguments:

`delay_display_seconds` a numeric value specifying the seconds to delay the display of the next image (It displays the images consecutively). This parameter applies only if the *display_all* is set to FALSE (spixel_masks_show method)

`display_all` a boolean. If TRUE then all images will be displayed in a grid (spixel_masks_show method)

`margin_btw_plots` a float number specifying the margins between the plots if the *display_all* parameter is set to TRUE (spixel_masks_show method)

`verbose` a boolean. If TRUE then information will be printed in the console (spixel_masks_show method)

Method `spixel_clusters_show()`**:**

Usage:

```
Image_Segmentation$spixel_clusters_show(
  spix_labels,
  color_palette = grDevices::rainbow,
  parameter_list_png = NULL
)
```

Arguments:

`spix_labels` a matrix. I can retrieve the "spix_labels" parameter by setting the "return_labels_2_dimensionsional" parameter to TRUE in the "spixel_segmentation" method (spixel_clusters_show method)

`color_palette` one of the color palettes. Use ?grDevices::topo.colors to see the available color palettes

`parameter_list_png` either NULL or a list of parameters passed to the ?grDevices::png function, such as list(filename = 'img.png', width = 100, height = 100, units = "px", pointsize = 12, bg = "white", type = "quartz")

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

https://pdfs.semanticscholar.org/6533/654973054b742e725fd433265700c07b48a2.pdf , "Image Segmentation using SLIC Superpixels and Affinity Propagation Clustering", Bao Zhou, 2013, International Journal of Science and Research (IJSR)

Examples

```
library(SuperpixelImageSegmentation)
```

```
path = system.file("images", "BSR_bsds500_image.jpg", package = "SuperpixelImageSegmentation")

im = OpenImageR::readImage(path)

init = Image_Segmentation$new()

num_spix = 10           # for illustration purposes
# num_spix = 600        # recommended number of superpixels

spx = init$spixel_segmentation(input_image = im,
                               superpixel = num_spix,
                               AP_data = TRUE,
                               use_median = TRUE,
                               return_labels_2_dimensionsional = TRUE,
                               sim_color_radius = 10)


#.......................
# plot the superpixel labels
#.......................

plt = init$spixel_clusters_show(spix_labels = spx$spix_labels,
                                color_palette = grDevices::rainbow,
                                parameter_list_png = NULL)

# plt


#.............................................
# create a binary image for a specified cluster label
#.............................................

pix_values = spx$spix_labels

target_cluster = 3                          # determine clusters visually ('plt' variable)

pix_values[pix_values != target_cluster] = 0     # set all other values to 0 (background)
pix_values[pix_values == target_cluster] = 1    # set the target_cluster to 1 (binary image)

# OpenImageR::imageShow(pix_values)
```

Index