Package: RsimMosaic (via r-universe)

October 25, 2024

Type Package

Title R Simple Image Mosaic Creation Library

Version 1.0.3

Date 2017-04-03

Author Alberto Krone-Martins

Maintainer Alberto Krone-Martins <algol@sim.ul.pt>

Description Provides a way to transform an image into a mosaic composed from a set of smaller images (tiles). It also contains a simple function for creating the tiles from a folder of images directly through R, without the need of any external code. At this moment only the JPEG format is supported, either as input (image and tiles) or output (mosaic transformed image).

License GPL (>= 2)

Depends R (>= 3.1.0)

Imports jpeg, fields, RANN

NeedsCompilation no

Date/Publication 2017-04-03 20:19:14 UTC

Repository https://algolkm.r-universe.dev

RemoteUrl https://github.com/cran/RsimMosaic

RemoteRef HEAD

RemoteSha d53e6ec362adc858274dcc230582d0e493d03740

Contents

`RsimMosaic-package` *R Simple IMage Mosaic creation library*

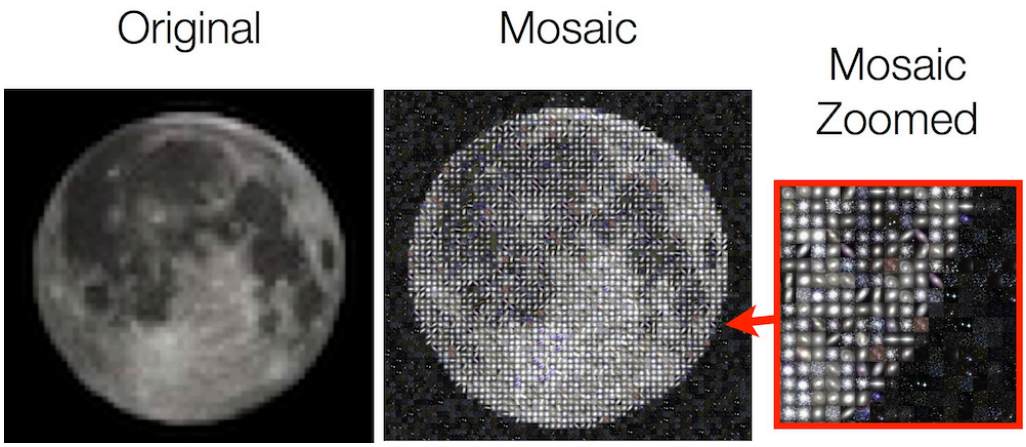
Description

RsimMosaic is a package for transforming an image into a mosaic composed from a set of smaller images (tiles). This library also contains a simple function for creating the tiles from a folder of images directly through R, without the need of any external code. At this moment only the JPEG format is supported, either as input (image and tiles) or output (mosaic transformed image).

Details

Package:	RsimMosaic
Type:	Package
Version:	1.0.2
Date:	2014-08-23
License:	GPL (>= 2)

The RsimMosaic, or the R Simple IMage Mosaic creation library, is a package for the production of mosaics. From a set of images that are called tiles, the implemented methods are able to compose a mosaic that mimics a user selected image. One example is represented in the figure below:



Note that the larger and the more varied is the adopted tile library, the more faithful will be the resulting mosaic. This package comes with a very small library, containing only 100 stamps of astronomical sources (called Messier objects) from the 2MASS catalogue, for example and test purposes.

Author(s)

Author: Alberto Krone-Martins

Maintainer: Alberto Krone-Martins <algol@sim.ul.pt>

See Also

composeMosaicFromImageRandom, composeMosaicFromImageRegular, createTiles

Examples

```
# This example will transform an image of the Moon into a mosaic
# composed by objects from the Messier catalogue. The tiles were
# created from the images of the 2MASS catalogue.
#
# Set the filename of the original image
#origImgFileN <- system.file("extdata", "verySmallMoon.jpg", package="RsimMosaic")
origImgFileN <- system.file("extdata", "reallyVerySmallMoon.jpg", package="RsimMosaic")

# Set the folder where the tiles library is located
pathToTileLib <- system.file("extdata/2Massier", package="RsimMosaic")

# Set the filename of the output image (the mosaic!)
outImgFileN <- file.path(tempdir(), "verySmallMoon-2MASS-Mosaic.jpg")

# Create the mosaic
composeMosaicFromImageRandom(origImgFileN, outImgFileN, pathToTileLib, removeTiles=FALSE)
```

---

addBackTile *Add a tile back to the tile library*

Description

This is a simple function to add a tile (with the filename `tileFilename`) back to the tile library (passed as the argument `libForMosaic`). The tile data in the parameter space is copied from the original tile library (passed as the argument `libForMosaicFull`).

Usage

```
addBackTile(tileFilename, libForMosaic, libForMosaicFull)
```

Arguments

`tileFilename`	The filename of the tile to add back in the tile library.
`libForMosaic`	The tile library where the tile should be added back.
`libForMosaicFull`	The original tile library containing the data of the all the tiles in the parameter space.

Value

It returns the tile library libForMosaic, with the requested tile added.

Author(s)

Alberto Krone-Martins

See Also

`removeTile`

Examples

```
# Creates the tile library data frame from the example tiles
my2MassTilesFull <- createLibraryIndexDataFrame(system.file("extdata/2Massier/",
 package="RsimMosaic"))
my2MassTiles <- my2MassTilesFull

# Get a random filename of one of the tiles
idx <- round(runif(1, 1, length(my2MassTiles[,1])))
tileFilename <- as.character(my2MassTiles[idx,1])

# Remove it from the library
my2MassTiles <- removeTile(tileFilename, my2MassTiles)

# Now, put it back
my2MassTiles <- addBackTile(tileFilename, my2MassTiles, my2MassTilesFull)
```

`bilinearInterpolator` *A function to perform bilinar interpolation*

Description

This function is just a wrapper to the `interp.surface.grid` function to perform bilinar interpolation of a regular matrix.

Usage

```
bilinearInterpolator(oldMatrix, pointsInNewX, pointsInNewY)
```

Arguments

oldMatrix	The original matrix.
pointsInNewX	The number of points in the new matrix (the number of new rows).
pointsInNewY	The number of points in the new matrix (the number of new columns).

Value

It returns an interpolated matrix. The size of the new matrix is pointsInNewX rows and pointsInNewY columns.

Author(s)

Alberto Krone-Martins

See Also

interp.surface.grid

Examples

```
library('jpeg')

# Read the R logo
logo <- readJPEG(system.file("img", "Rlogo.jpg", package="jpeg"))

# Create a scaled down version of the R channel
intrpArray <- array(dim=c(20, 20, 3))
intrpArray[,,1] <- bilinearInterpolator(logo[,,1], dim(intrpArray)[1], dim(intrpArray)[2])

# Display the results
dev.new()
image(logo[,,1], main="Original")
dev.new()
image(intrpArray[,,1], main="Scaled down")
```

---

composeMosaicFromImageRandom

Randomly transform an image into a mosaic

Description

A function to compose the mosaic of an image based on regular tiles. This function will compute the mosaic by randomly replacing the pixels of the original image with tiles from a tile library.

Usage

```
composeMosaicFromImageRandom(originalImageFileName, outputImageFileName,
  imagesToUseInMosaic, useGradients = FALSE, removeTiles = TRUE,
  fracLibSizeThreshold = 0.7, repFracSize = 0.25, verbose = TRUE)
```

Arguments

`originalImageFileName`
: The original image you want to use to create the mosaic from (note that for the sake of your computer's memory, this image should be small, about 128 or 256 pixels wide, or so...).

`outputImageFileName`
: The filename (with the path) where you want the image to be stored. This image can be quite large, depending on the number of pixels in the original image and on the tiles.

`imagesToUseInMosaic`
: A path with the folder where the tiles are contained (note that the tiles should be square and for the sake of your computer's memory, small, e.g. 40x40, 120x120 pixels or so...). You can use the function `createTiles` to create a folder with smaller tiles from a folder with your original images.

`useGradients`
: A flag to indicate if approximate gradients should be taken into account when selecting the tiles.

`removeTiles`
: A flag to indicate if the user wants to temporarily remove tiles from the library after using them, so tile repetition would be discoraged by the software. If the tile library is small, the tiles will be added back to the library after its size reaches a certain threshold (defined by `fracLibSizeThreshold`).

`fracLibSizeThreshold`
: The fraction of the size of the original tile library when the tiles must be (randomly) put back into the library.

`repFracSize`
: The fraction of the size of the original tile library to replace when filling back the array (it should be smaller than, or equal to, 1-fracLibSizeThreshold).

`verbose`
: A flag indicating if the user wants to have messages during the mosaic creation.

Author(s)

Alberto Krone-Martins

See Also

`composeMosaicFromImageRegular`

Examples

```
# Set the filename of the original image
#origImgFileN <- system.file("extdata", "verySmallMoon.jpg", package="RsimMosaic")
origImgFileN <- system.file("extdata", "reallyVerySmallMoon.jpg", package="RsimMosaic")

# Set the folder where the tiles library is located
pathToTileLib <- system.file("extdata/2Massier", package="RsimMosaic")

# Set the filename of the output image (the mosaic!)
outImgFileN <- file.path(tempdir(), "verySmallMoon-2MASS-Mosaic.jpg")

# Create the mosaic
```

```
composeMosaicFromImageRandom(origImgFileN, outImgFileN, pathToTileLib, removeTiles=FALSE)
```

---

composeMosaicFromImageRandomOptim *Randomly transform an image into a mosaic optimizing the library usage*

Description

A function to compose the mosaic of an image based on regular tiles. This function will compute the mosaic by randomly replacing the pixels of the original image with tiles from a tile library, and it will try to use as many tiles from the library as possible.

Usage

```
composeMosaicFromImageRandomOptim(originalImageFileName, outputImageFileName,
  imagesToUseInMosaic, useGradients = FALSE, removeTiles = TRUE,
  fracLibSizeThreshold = 0.7, repFracSize = 0.25, verbose = TRUE, neig=20)
```

Arguments

`originalImageFileName`
: The original image you want to use to create the mosaic from (note that for the sake of your computer's memory, this image should be small, about 128 or 256 pixels wide, or so...).

`outputImageFileName`
: The filename (with the path) where you want the image to be stored. This image can be quite large, depending on the number of pixels in the original image and on the tiles.

`imagesToUseInMosaic`
: A path with the folder where the tiles are contained (note that the tiles should be square and for the sake of your computer's memory, small, e.g. 40x40, 120x120 pixels or so...). You can use the function `createTiles` to create a folder with smaller tiles from a folder with your original images.

`useGradients`
: A flag to indicate if approximate gradients should be taken into account when selecting the tiles.

`removeTiles`
: A flag to indicate if the user wants to temporarily remove tiles from the library after using them, so tile repetition would be discoraged by the software. If the tile library is small, the tiles will be added back to the library after its size reaches a certain threshold (defined by `fracLibSizeThreshold`).

`fracLibSizeThreshold`
: The fraction of the size of the original tile library when the tiles must be (randomly) put back into the library.

`repFracSize`
: The fraction of the size of the original tile library to replace when filling back the array (it should be smaller than, or equal to, 1-fracLibSizeThreshold).

`verbose`
: A flag indicating if the user wants to have messages during the mosaic creation.

`neig`
: An integer indicating the number of neighbors to retrieve from the tile library.

Author(s)

Alberto Krone-Martins

See Also

composeMosaicFromImageRandom

Examples

```
# Set the filename of the original image
#origImgFileN <- system.file("extdata", "verySmallMoon.jpg", package="RsimMosaic")
origImgFileN <- system.file("extdata", "reallyVerySmallMoon.jpg", package="RsimMosaic")

# Set the folder where the tiles library is located
pathToTileLib <- system.file("extdata/2Massier", package="RsimMosaic")

# Set the filename of the output image (the mosaic!)
outImgFileN <- file.path(tempdir(), "verySmallMoon-2MASS-Mosaic.jpg")

# Create the mosaic
composeMosaicFromImageRandomOptim(origImgFileN, outImgFileN, pathToTileLib, removeTiles=FALSE)
```

---

composeMosaicFromImageRegular

Regularly transform an image into a mosaic

Description

A function to compose the mosaic of an image based on regular tiles. This function will compute the mosaic by regularly replacing the pixels of the original image with tiles from a tile library.

Usage

```
composeMosaicFromImageRegular(originalImageFileName, outputImageFileName,
  imagesToUseInMosaic, useGradients = FALSE, removeTiles = TRUE,
  fracLibSizeThreshold = 0.7, repFracSize = 0.25, verbose = TRUE)
```

Arguments

`originalImageFileName`
: The original image you want to use to create the mosaic from (note that for the sake of your computer's memory, this image should be small, about 128 or 256 pixels wide, or so...).

`outputImageFileName`
: The filename (with the path) where you want the image to be stored. This image can be quite large, depending on the number of pixels in the original image and on the tiles.

`imagesToUseInMosaic`
: A path with the folder where the tiles are contained (note that the tiles should be square and for the sake of your computer's memory, small, e.g. 40x40, 120x120 pixels or so...). You can use the function `createTiles` to create a folder with smaller tiles from a folder with your original images.

`useGradients`
: A flag to indicate if approximate gradients should be taken into account when selecting the tiles.

`removeTiles`
: A flag to indicate if the user wants to temporarily remove tiles from the library after using them, so tile repetition would be discoraged by the software. If the tile library is small, the tiles will be added back to the library after its size reaches a certain threshold (defined by `fracLibSizeThreshold`).

`fracLibSizeThreshold`
: The fraction of the size of the original tile library when the tiles must be (randomly) put back into the library.

`repFracSize`
: The fraction of the size of the original tile library to replace when filling back the array (it should be smaller than, or equal to, 1-fracLibSizeThreshold).

`verbose`
: A flag indicating if the user wants to have messages during the mosaic creation.

Author(s)

Alberto Krone-Martins

See Also

`composeMosaicFromImageRandom`

Examples

```
# Set the filename of the original image
#origImgFileN <- system.file("extdata", "verySmallMoon.jpg", package="RsimMosaic")
origImgFileN <- system.file("extdata", "reallyVerySmallMoon.jpg", package="RsimMosaic")

# Set the folder where the tiles library is located
pathToTileLib <- system.file("extdata/2Massier", package="RsimMosaic")

# Set the filename of the output image (the mosaic!)
outImgFileN <- file.path(tempdir(), "verySmallMoon-2MASS-Mosaic.jpg")

# Create the mosaic
composeMosaicFromImageRegular(origImgFileN, outImgFileN, pathToTileLib, removeTiles=TRUE)
```

`computeStatisticalQuantitiesPixel`
A function to compute the pixel data in a certain parameter space

---

Description

A function to compute the relevant pixel quantity (the RGB color). Optionally it can also output the values of relevant, nearby pixels as RGB colors at the Left, UpperLeft, Upper, UpperRight, Right, LowerRight, Lower, and LowerLeft pixels.

Usage

```
computeStatisticalQuantitiesPixel(i, j, img, useGradients = FALSE)
```

Arguments

i	The row (or the X in the image).
j	The column (or the Y in the image).
img	An image array (as created by the readJPEG function from the jpeg library).
useGradients	A flag indicating if the values of the nearby pixels should be returned.

Value

An array with the relevant pixel quantites.

Author(s)

Alberto Krone-Martins

See Also

computeStatisticalQuantitiesTile

Examples

```
# Read the R logo and output the value of its pixel (50, 5) in the parameter space
library('jpeg')
logo <- readJPEG(system.file("img", "Rlogo.jpg", package="jpeg")) # Read the R logo
computeStatisticalQuantitiesPixel(50, 5, logo) # Compute the quantities at the pixel (5,5)
```

---

computeStatisticalQuantitiesTile

A function to compute the tile data in a certain parameter space

---

Description

A function to compute the relevant statistical quantities (only the median value is implemented) of the RGB colors for the entire image. Optionally it can also compute the median values of the RGB colors at the Left, UpperLeft, Upper, UpperRight, Right, LowerRight, Lower, and LowerLeft, corners of the image.

Usage

```
computeStatisticalQuantitiesTile(img, useGradients = FALSE)
```

Arguments

img	An image array (as created by the readJPEG function from the jpeg library).
useGradients	A flag indicating if the values of the relevant quantities should be calculated in the corners of the image.

Details

The data is defined as the median RGB colors of the tile images at this version. Optionally, the median RGB values of the tile image corners are also calculated.

Value

An array with the relevant quantities calculated.

Author(s)

Alberto Krone-Martins

See Also

readJPEG, computeStatisticalQuantitiesPixel

Examples

```
# Read the R logo and output its values in the parameter space
library('jpeg')
logo <- readJPEG(system.file("img", "Rlogo.jpg", package="jpeg")) # Read the R logo
computeStatisticalQuantitiesTile(logo) # Compute the quantities
```

---

createLibraryIndexDataFrame

A function to create the tile library data frame

Description

A function to create the tile library data frame containing the data of the tiles in the parameter space, as well as the filename of the tiles.

Usage

```
createLibraryIndexDataFrame(path, saveLibraryIndex = FALSE,
  libraryFilename, useGradients = FALSE)
```

Arguments

path	A path indicating where the images that will compose the library are stored.
saveLibraryIndex	A flag to indicate if the library should be saved in a file.
libraryFilename	The filename to use if the user wants to store the library to a file.
useGradients	A flag indicating if the values of the data in the parameter space in the corners of the tile images should be calculated.

Details

The tile image data in the parameter space is calculated by computeStatisticalQuantitiesTile.

Value

It returns a data frame containing the filename of each tile found at the path and the median RGB values of each tile. Optionally it also includes the median RGB values of the corners of the image.

Author(s)

Alberto Krone-Martins

See Also

computeStatisticalQuantitiesTile

Examples

```
# Creates the tile library data frame from the example tiles
my2MassTiles <- createLibraryIndexDataFrame(system.file("extdata/2Massier/",
 package="RsimMosaic"))
```

---

createTiles *Tile creation from a folder of images*

Description

A very simple function to create tiles from a folder containing JPEG images. It uses bilinear interpolation (via the bilinearInterpolator function), thus note that the quality of the tiles will be sub-optimal. Thus, for high quality purposes please use another external tool providing better interpolation schemes to create your tiles (bicubic splines, sincz, ...).

Usage

```
createTiles(inPath, outPath, tileHeight = 40, verbose = TRUE)
```

Arguments

inPath	A path with the folder where the images are contained.
outPath	A path with the folder where the tiles will be created (if the folder does not exists, it will be created).
tileHeight	The tile height in pixels.
verbose	A boolean flag to indicate if the user wants to have screen output or not.

Author(s)

Alberto Krone-Martins

See Also

bilinearInterpolator

Examples

```
# Set the folder where the original images are located
pathToOriginalImages <- system.file("extdata/2Massier/", package="RsimMosaic")

# Set the folder where the tiles will be stored. It will be created if it does not exists.
pathToTileImages <- paste(tempdir(), "/myTiles/", sep="")

# Create the tiles (10 pix are used just to make this example run fast)
createTiles(pathToOriginalImages, pathToTileImages, tileHeight=10)
```

---

getCloseMatch *Get a tile which is a close match for a pixel in the parameter space*

Description

This function will return the filename of a tile that is a close match to a pixel in the parameter space. The nneig matches are selected using a nearest neighbour search (nn2) in the tile library (libraryDataFrame). After the candidates selected, one of the is randomly choosen and its filename is returned by the function.

Usage

```
getCloseMatch(pixelArray, libraryDataFrame, nneig = 20)
```

Arguments

pixelArray	The parameters of the pixel to get a similar image from the library in the parameter space.
libraryDataFrame	The tile library containing the data of the tiles in the parameter space.
nneig	Number of neighbours to retrieve in the intermediate test. Only one of the neighbours will be returned to the user.

Value

The filename of a tile that is a close match to a pixel in the parameter space.

Author(s)

Alberto Krone-Martins

See Also

nn2

Examples

```
# Creates the tile library data frame from the example tiles
my2MassTiles <- createLibraryIndexDataFrame(system.file("extdata/2Massier", package="RsimMosaic"))

# Get a close match for the point with parameters (0.2, 0.3, 0.2)
getCloseMatch(c(0.2, 0.3, 0.2), my2MassTiles)

# Get another close match for the point with parameters (0.2, 0.3, 0.2)
getCloseMatch(c(0.2, 0.3, 0.2), my2MassTiles)
```

---

removeTile *Remove a tile from the tile library*

Description

This is a simple function to remove a tile (with the filename `tileFilename`) from the tile library (passed as the argument `libForMosaic`).

Usage

```
removeTile(tileFilename, libForMosaic)
```

Arguments

`tileFilename`	The filename of the tile to remove from the tile library.
`libForMosaic`	The library containing the data of the tiles in the parameter space from which the tile should be removed.

Value

It returns the tile library libForMosaic, with the requested tile removed.

Author(s)

Alberto Krone-Martins

See Also

addBackTile

Examples

```
# Creates the tile library data frame from the example tiles
my2MassTiles <- createLibraryIndexDataFrame(system.file("extdata/2Massier", package="RsimMosaic"))

# Get a random filename of one of the tiles
idx <- round(runif(1, 1, length(my2MassTiles[,1])))
tileToRemove <- as.character(my2MassTiles[idx,1])

# Remove it from the library
my2MassTiles <- removeTile(tileToRemove, my2MassTiles)
```

Index