Package ‘countcolors’

July 22, 2025

Type Package

Title Locates and Counts Pixels Within Color Range(s) in Images

Version 0.9.1

Maintainer Hannah Weller <hannahiweller@gmail.com>

Description Counts colors within color range(s) in images, and provides a masked version of the image with targeted pixels changed to a different color. Output includes the locations of the pixels in the images, and the proportion of the image within the target color range with optional background masking. Users can specify multiple color ranges for masking.

Imports colordistance, tools, graphics, png, jpeg

Depends R (>= 3.4.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

Suggests knitr, rmarkdown, scatterplot3d

VignetteBuilder knitr

NeedsCompilation no

Author Hannah Weller [aut, cre]

Repository CRAN

Date/Publication 2019-01-12 22:20:53 UTC

Contents

changePixelColor *Change all specified pixels to a new color*

Description

Changes pixels in an image to a different color and displays and/or returns the re-colored image.

Usage

```
changePixelColor(pixel.array, pixel.idx, target.color = "green",
  return.img = FALSE, plotting = TRUE, main = "")
```

Arguments

`pixel.array`	An image represented as a 3D array (as read in by `readJPEG`, `readPNG`, or `loadImage`) in which to change pixel colors.
`pixel.idx`	An n x 2 matrix of index coordinates specifying which pixels to change, where rows are pixels and columns are X and Y coordinates in the image.
`target.color`	Color with which to replace specified pixels. Can be either a an RGB triplet or one of the colors listed by `colors`.
`return.img`	Logical. Should RGB array with swapped colors be returned?
`plotting`	Logical. Should output be plotted in the plot window?
`main`	Optional title to display for image.

Value

Raster array with indicated pixels changed to target color, if `return.img = TRUE`.

Examples

```
# Change a rectangle in the center to black
flowers <- jpeg::readJPEG(system.file("extdata", "flowers.jpg", package =
"countcolors"))

sinister.object <- expand.grid(c(114:314), c(170:470))

countcolors::changePixelColor(flowers, sinister.object, target.color = "black")

## Not run:
# Change all the white flowers to magenta
indicator.img <- countcolors::sphericalRange(flowers, center = c(1, 1, 1),
radius = 0.1, color.pixels = TRUE, plotting = FALSE)

countcolors::changePixelColor(flowers, indicator.img$pixel.idx,
target.color="magenta")

## End(Not run)
```

countColors — *Count the number of pixels within a color range or ranges*

Description

Counts the pixels within a color range; ranges can be spherical (see `sphericalRange`) or rectangular (`rectangularRange`). If multiple ranges are specified, each one is colored using a different indicator color.

Usage

```
countColors(path, color.range = "spherical", center, radius, lower,
  upper, bg.lower = rep(0.8, 3), bg.upper = rep(1, 3),
  target.color = c("magenta", "cyan", "yellow"), plotting = FALSE,
  save.indicator = FALSE, dpi = 72, return.indicator = FALSE)
```

Arguments

`path`	Path to the image (JPEG or PNG).
`color.range`	Type of range being specified. Must be either "spherical" or "rectangular".
`center`	A vector or n x 3 matrix of color centers (RGB triplets) around which to search using spherical color range. RGB range 0-1 (not 0-255). See details.
`radius`	Values between 0 and 1 specifying the size of the area around `center` to search. The same number of centers and radii must be specified.
`lower, upper`	RGB triplet(s) specifying the bounds of color space to search. Must be the same length. See details.
`bg.lower, bg.upper`	RGB triplets specifying the bounds of color space to ignore as background, or `NULL` to use the entire image.
`target.color`	If an indicator image is created, the color with which to replace specified pixels. Can be either an RGB triplet or one of the colors listed by `colors`.
`plotting`	Logical. Should output be plotted in the plot window?
`save.indicator`	Logical OR path for saving indicator image. If TRUE, saves image to the same directory as the original image as 'originalimagename_masked.png'; if a path is provided, saves it to that directory/name instead.
`dpi`	Resolution (dots per image) for saving indicator image.
`return.indicator`	Logical. Should an indicator image (RGB array with targeted pixels changed to indicator color) be returned?

Details

More than one set of ranges can be specified for the color search space, but the same number of arguments must be provided in each case (so the number of pixels and centers must be the same if using a spherical range, and the number of upper and lower bounds must be the same if using a rectangular one).

For `center`, `upper`, and `lower`, which call for RGB triplets, provide either a vector of RGB triplets in RGB order, i.e. c(R1, G1, B1, R2, G2, B2, etc) or a 3-column matrix with one row per RGB triplet. If a vector is provided, it is coerced to a 3-column matrix, and must therefore be a multiple of 3.

Value

A list with the following attributes:

- `pixel.idx`: Unique coordinates of pixels within color range(s).
- `pixel.fraction`: Proportion of the non-background image within color range(s), found by dividing the number of pixels in `pixel.idx` by the number of non-background pixels in the image.
- `indicator.img`: If `return.indicator` = `TRUE`, RGB array with color-swapped pixels.

See Also

`countColorsInDirectory`

Examples

```
# Try out a few different radii for white pelicans
pelicans <- system.file("extdata", "pelicans.jpg", package = "countcolors")

white.ctr <- rep(0.9, 9)
white.radii <- c(0.5, 0.3, 0.1)

# Magenta = 50% threshold, cyan = 30% threshold, yellow = 10% threshold
pelican.example <- countcolors::countColors(pelicans, center = white.ctr,
radius = white.radii, bg.lower = NULL, plotting = TRUE)
```

`countColorsInDirectory`

Count colors within range(s) in every image in a directory

Description

A wrapper for `countColors` that finds every image (JPEG or PNG) in a #' folder and counts colors in each image.

Usage

```
countColorsInDirectory(folder, color.range = "spherical", center, radius,
  lower, upper, bg.lower = rep(0.8, 3), bg.upper = rep(1, 3),
  target.color = c("magenta", "cyan", "yellow"), plotting = FALSE,
  save.indicator = FALSE, dpi = 72, return.indicator = FALSE)
```

Arguments

`folder`	Path to a folder containing images.
`color.range`	Type of range being specified. Must be either "spherical" or "rectangular".
`center`	A vector or n x 3 matrix of color centers (RGB triplets) around which to search using spherical color range. RGB range 0-1 (not 0-255). See details.
`radius`	Values between 0 and 1 specifying the size of the area around `center` to search. The same number of centers and radii must be specified.
`lower`	RGB triplet(s) specifying the bounds of color space to search. Must be the same length. See details.
`upper`	RGB triplet(s) specifying the bounds of color space to search. Must be the same length. See details.
`bg.lower`	RGB triplets specifying the bounds of color space to ignore as background, or `NULL` to use the entire image.
`bg.upper`	RGB triplets specifying the bounds of color space to ignore as background, or `NULL` to use the entire image.
`target.color`	If an indicator image is created, the color with which to replace specified pixels. Can be either an RGB triplet or one of the colors listed by `colors`.
`plotting`	Logical. Should output be plotted in the plot window?
`save.indicator`	Logical OR path for saving indicator image. If TRUE, saves image to the same directory as the original image as 'originalimagename_masked.png'; if a path is provided, saves it to that directory/name instead.
`dpi`	Resolution (dots per image) for saving indicator image.
`return.indicator`	Logical. Should an indicator image (RGB array with targeted pixels changed to indicator color) be returned?

Value

A list of `countColors` lists, one for each image.

See Also

`countColors`

Examples

```
## Not run:
folder <- system.file("extdata", package = "countcolors")

# Screen out white in both the flower image and the pelican image
upper <- c(1, 1, 1)
lower <- c(0.8, 0.8, 0.8)

white.screen <- countcolors::countColorsInDirectory(folder, color.range =
"rectangular", upper = upper, lower = lower, bg.lower = NULL, plotting =
TRUE, target.color = "turquoise")

## End(Not run)
```

`plotArrayAsImage` *Plot a 3D array as an RGB image*

Description

Plots a 3D array as an RGB image in the plot window.

Usage

```
plotArrayAsImage(rgb.array, main = "")
```

Arguments

`rgb.array`	3D RGB array with R, G, and B channels (pixel rows x pixel columns x color channels) to be plotted in the plot window as an actual image.
`main`	Optional title to display for image.

Examples

```
# Read in image
flowers <- jpeg::readJPEG(system.file("extdata", "flowers.jpg", package =
"countcolors"))

# Plot
plotArrayAsImage(flowers, main = "flowers!")
```

`rectangularRange` *Find pixels within a target color range defined by boundaries in each channel*

Description

Searches for pixels within a set of upper and lower limits for each color channel. Essentially draws a 'box' around a region of color space in which to search for pixels.

Usage

```
rectangularRange(pixel.array, upper, lower, target.color = "green",
  main = "", color.pixels = TRUE, plotting = TRUE)
```

Arguments

`pixel.array`	An image represented as a 3D array (as read in by `readJPEG`, `readPNG`, or `loadImage`) in which to change pixel colors.
`upper, lower`	RGB triplet specifying the bounds of color space search. See details.
`target.color`	Color with which to replace specified pixels. Can be either an RGB triplet or one of the colors listed by `colors`.
`main`	Optional title to display for image.
`color.pixels`	Logical. Should a diagnostic image with pixels changed to `target.color` be returned?
`plotting`	Logical. Should output be plotted in the plot window?

Details

`lower` and `upper` should be vectors of length 3 in a 0-1 range, in the order R-G-B. For example, the upper bounds for white would be c(1, 1, 1), and the lower bounds might be c(0.8, 0.8, 0.8). This would search for all pixels where the red value, blue value, AND green value are all between 0.8 and 1.

Value

A list with the following elements:

- `pixel.idx`: Coordinates of pixels within color range.
- `img.fraction`: Proportion of the image within color range.
- `original.img`: The original RGB array.
- `indicator.img`: If `color.pixels` = `TRUE`, RGB array with color-swapped pixels.

Examples

```
flowers <- jpeg::readJPEG(system.file("extdata", "flowers.jpg", package =
"countcolors"))

# Define upper and lower bounds for white
lower <- rep(0.8, 3)
upper <- rep(1, 3)

white.flowers <- countcolors::rectangularRange(flowers, rep(1, 3), rep(0.85,
3), target.color = "turquoise")
```

`sphericalRange` *Find pixels within a target color range defined by a center and search radius*

Description

Searches for pixels within a radius of a single color. Draws a sphere around a region of color space in which to search for pixels.

Usage

```
sphericalRange(pixel.array, center, radius, target.color = "green",
  main = "", color.pixels = TRUE, plotting = TRUE)
```

Arguments

`pixel.array`	An image represented as a 3D array (as read in by `readJPEG`, `readPNG`, or `loadImage`) in which to change pixel colors.
`center`	A single color (RGB triplet) around which to search. RGB range 0-1 (not 0-255).
`radius`	A value between 0 and 1 specifying the size of the area around `center` to search.
`target.color`	Color with which to replace specified pixels. Can be either a an RGB triplet or one of the colors listed by `colors`.
`main`	Optional title to display for image.
`color.pixels`	Logical. Should a diagnostic image with pixels changed to `target.color` be returned?
`plotting`	Logical. Should output be plotted in the plot window?

Details

`lower` and `upper` should be vectors of length 3 in a 0-1 range, in the order R-G-B. For example, the upper bounds for white would be c(1, 1, 1), and the lower bounds might be c(0.8, 0.8, 0.8). This would search for all pixels where the red value, blue value, AND green value are all between 0.8 and 1.

Value

A list with the following elements:

- `pixel.idx`: Coordinates of pixels within color range.
- `img.fraction`: Proportion of the image within color range.
- `original.img`: The original RGB array.
- `indicator.img`: If `color.pixels` = `TRUE`, RGB array with color-swapped pixels.

Examples

```
# Target color: change all of the red flowers to turquoise
flowers <- jpeg::readJPEG(system.file("extdata", "flowers.jpg", package =
"countcolors"))

# Red:
center <- c(255, 75, 75) / 255

# Setting the radius too low:
red.flowers <- countcolors::sphericalRange(flowers, center = center, radius =
0.05, target.color = "turquoise")

# Setting the radius too high:
red.flowers <- countcolors::sphericalRange(flowers, center = center, radius =
0.4, target.color = "turquoise")

# Setting the radius just right:
red.flowers <- countcolors::sphericalRange(flowers, center = center, radius =
0.2, target.color = "turquoise")
```

Index