Package 'rayimage'

June 28, 2020

Type Package

Title Image Processing for Simulated Cameras

Version 0.3.1

Author Tyler Morgan-Wall

Maintainer Tyler Morgan-Wall <tylermw@gmail.com>

Description Uses convolution-based techniques to generate simulated camera bokeh, depth of field, and other camera effects, using an image and an optional depth map. Accepts both filename inputs and in-memory array representations of images and matrices. Includes functions to perform 2D convolutions, reorient and resize images/matrices, add image overlays, generate camera vignette effects, and add titles to images.

License GPL-3

LazyData true

Depends R (>= 3.0.2)

Imports Rcpp, raster, png, magrittr, jpeg, grDevices

Suggests magick

LinkingTo Rcpp, RcppArmadillo, progress

RoxygenNote 7.1.0

URL https://www.rayimage.dev, https://github.com/tylermorganwall/rayimage

BugReports https://github.com/tylermorganwall/rayimage/issues

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-06-28 20:00:02 UTC

R topics documented:

`add_image_overlay` *Add Overlay*

Description

Takes an RGB array/filename and adds an image overlay.

Usage

```
add_image_overlay(
  image,
  image_overlay = NULL,
  rescale_original = FALSE,
  alpha = NULL,
  filename = NULL,
  preview = FALSE
)
```

Arguments

`image`	Image filename or 3-layer RGB array.
`image_overlay`	Default `NULL`. Either a string indicating the location of a png image to overlay over the image (transparency included), or a 4-layer RGBA array. This image will be resized to the dimension of the image if it does not match exactly.
`rescale_original`	Default `FALSE`. If `TRUE`, function will resize the original image to match the overlay.
`alpha`	Default `NULL`, using overlay's alpha channel. Otherwise, this sets the alpha transparency by multiplying the existing alpha channel by this value (between 0 and 1).
`filename`	Default `NULL`. File to save the image to. If `NULL` and `preview = FALSE`, returns an RGB array.
`preview`	Default `FALSE`. If `TRUE`, it will display the image in addition to returning it.

Value

3-layer RGB array of the processed image.

Examples

```
#Plot the dragon
plot_image(dragon)

#Add an overlay of a red semi-transparent circle:
circlemat = generate_2d_disk(min(dim(dragon)[1:2]))
circlemat = circlemat/max(circlemat)

#Create RGBA image, with a transparency of 0.5
rgba_array = array(1, dim=c(nrow(circlemat),ncol(circlemat),4))
rgba_array[,,1] = circlemat
rgba_array[,,2] = 0
rgba_array[,,3] = 0
dragon_clipped = dragon
dragon_clipped[dragon_clipped > 1] = 1

add_image_overlay(dragon_clipped, image_overlay = rgba_array,
                  alpha=0.5, preview = TRUE)
```

---

add_title *Add Title*

Description

Takes an RGB array/filename and adds a title with an optional titlebar.

Usage

```
add_title(
  image,
  title_text = "",
  title_offset = c(20, 20),
  title_color = "black",
  title_size = 30,
  title_font = "sans",
  title_style = "normal",
  title_bar_color = NULL,
  title_bar_alpha = 0.5,
  title_position = "northwest",
  filename = NULL,
  preview = FALSE
)
```

Arguments

image	Image filename or 3-layer RGB array.
title_text	Default NULL. Text. Adds a title to the image, using magick::image_annotate.
title_offset	Default c(20,20). Distance from the top-left (default, gravity direction in image_annotate) corner to offset the title.
title_color	Default black. Font color.
title_size	Default 30. Font size in pixels.
title_font	Default sans. String with font family such as "sans", "mono", "serif", "Times", "Helvetica", "Trebuchet", "Georgia", "Palatino" or "Comic Sans".
title_style	Default normal. Font style (e.g. italic).
title_bar_color	Default NULL. If a color, this will create a colored bar under the title.
title_bar_alpha	Default 0.5. Transparency of the title bar.
title_position	Default northwest. Position of the title.
filename	Default NULL. File to save the image to. If NULL and preview = FALSE, returns an RGB array.
preview	Default FALSE. If TRUE, it will display the image in addition to returning it.

Value

3-layer RGB array of the processed image.

Examples

```
#Plot the dragon
add_title(dragon, preview = TRUE, title_text = "Dragon", title_size=20)

#That's hard to see--let's add a title bar:

add_title(dragon, preview = TRUE, title_text = "Dragon", title_size=20,
         title_bar_color="white")


#Change the width of the bar:

add_title(dragon, preview = TRUE, title_text = "Dragon", title_size=20,
         title_bar_color="white", title_offset = c(12,12))

#Change the color and title color:

add_title(dragon, preview = TRUE, title_text = "Dragon", title_size=20,
         title_bar_color="red", title_color = "white", title_offset = c(12,12))


#Change the transparency:
```

```
add_title(dragon, preview = TRUE, title_text = "Dragon", title_size=20, title_bar_alpha = 0.8,
         title_bar_color="red", title_color = "white", title_offset = c(12,12))
```

---

add_vignette *Add Vignette Effect*

Description

Takes an RGB array/filename and adds a camera vignette effect.

Usage

```
add_vignette(image, vignette = 0.5, filename = NULL, preview = FALSE)
```

Arguments

image	Image filename or 3-layer RGB array.
vignette	Default 0.5. A camera vignetting effect will be added to the image. 1 is the darkest vignetting, while 0 is no vignetting. If vignette is a length-2 vector, the second entry will control the blurriness of the vignette effect (1 is the default, e.g. 2 would double the blurriness but would take much longer to compute).
filename	Default NULL. Filename which to save the image. If NULL and preview = FALSE, returns an RGB array.
preview	Default FALSE. If TRUE, it will display the image in addition to returning it.

Value

3-layer RGB array of the processed image.

Examples

```
#Plot the dragon
plot_image(dragon)

#Add a vignette effect:

add_vignette(dragon, preview = TRUE, vignette = 0.5)

#Darken the vignette effect:

add_vignette(dragon, preview = TRUE, vignette = 1)


#Increase the width of the blur by 50%:

add_vignette(dragon, preview = TRUE, vignette = c(1,1.5))
```

dragon *Dragon Image*

Description

Dragon Image

Usage

```
dragon
```

Format

An RGB 3-layer HDR array with 200 rows and 200 columns, generated using the rayrender package.

dragondepth *Dragon Depthmap*

Description

Dragon Depthmap

Usage

```
dragondepth
```

Format

An matrix with 200 rows and 200 columns, representing the depth into the `dragon` image scene. Generated using the rayrender package. Distances range from 847 to 1411.

generate_2d_disk *Generate 2D Disk*

Description

Generates a 2D disk with a gradual falloff.

Disk generated using the following formula:

(-22.35 cos(1.68 r^2) + 85.91 sin(1.68 r^2)) exp(-4.89 r^2) + (35.91 cos(4.99 r^2) - 28.87 sin(4.99 r^2)) exp(-4.71 r^2) + (-13.21 cos(8.24 r^2) - 1.57 sin(8.24 r^2)) exp(-4.05 r^2) + (0.50 cos(11.90 r^2) + 1.81 sin(11.90 r^2)) exp(-2.92 r^2) + (0.13 cos(16.11 r^2) - 0.01 sin(16.11 r^2)) exp(-1.51 r^2)

The origin of the coordinate system is the center of the matrix.

Usage

```
generate_2d_disk(dim = 11)
```

Arguments

dim	Default 11. The dimensions of the resulting square matrix.

Examples

```
image(generate_2d_disk(101), asp=1)
```

generate_2d_exponential *Generate 2D exponential Distribution*

Description

Generates a 2D exponential distribution, with an optional argument to take the exponential to a user-defined power.

Usage

```
generate_2d_exponential(falloff = 1, dim = 11, width = 3)
```

Arguments

falloff	Default 1. Falloff of the exponential.
dim	Default 11. The dimensions of the resulting square matrix.
width	Default 3 (-10 to 10). The range in which to compute the distribution.

Examples

```
image(generate_2d_exponential(1,31,3), asp=1)
```

generate_2d_gaussian *Generate 2D Gaussian Distribution*

Description

Generates a 2D gaussian distribution, with an optional argument to take the gaussian to a user-defined power.

Usage

```
generate_2d_gaussian(sd = 1, power = 1, dim = 11, width = 3)
```

Arguments

sd	Default 1. Standard deviation of the normal distribution
power	Default 1. Power to take the distribution. Higher values will result in a sharper peak.
dim	Default 11. The dimensions of the resulting square matrix.
width	Default 3 (-10 to 10). The range in which to compute the distribution.

Examples

```
image(generate_2d_gaussian(1,1,31), asp=1)
```

interpolate_array *Matrix/Array Interpolation*

Description

Given a series of X and Y coordinates and an array/matrix, interpolates the Z coordinate using bilinear interpolation.

Usage

```
interpolate_array(image, x, y)
```

Arguments

image	Image filename, a matrix, or a 3-layer RGB array.
x	X indices (or fractional index) to interpolate.
y	Y indices (or fractional index) to interpolate.

Value

Either a vector of values (if image is a matrix) or a list of interpolated values from each layer.

Examples

```
#Interpolate a matrix
interpolate_array(volcano,c(10,10.1,11),c(30,30.5,33))
#Interpolate a 3-layer array (returns list for each channel)
interpolate_array(dragon,c(10,10.1,11),c(30,30.5,33))
```

---

plot_image *Plot Image*

Description

Displays the image in the current device.

Usage

```
plot_image(input, rotate = 0, keep_user_par = FALSE, ...)
```

Arguments

input	Image or filename of an image to be plotted.
rotate	Default 0. Rotates the output. Possible values: 0, 90, 180, 270.
keep_user_par	Default TRUE. Whether to keep the user's par() settings. Set to FALSE if you want to set up a multi-pane plot (e.g. set par(mfrow)).
...	Additional arguments to pass to the raster::plotRGB function that displays the map.

Examples

```
#Plot the dragon array
plot_image(dragon)
```

---

render_bokeh *Render Bokeh*

Description

Takes an image and a depth map to render the image with depth of field (i.e. similar to "Portrait Mode" in an iPhone). User can specify a custom bokeh shape, or use one of the built-in bokeh types.

Usage

```
render_bokeh(
  image,
  depthmap,
  focus = 0.5,
  focallength = 100,
  fstop = 4,
  filename = NULL,
  preview = TRUE,
  preview_focus = FALSE,
  bokehshape = "circle",
  bokehintensity = 1,
  bokehlimit = 0.8,
  rotation = 0,
  aberration = 0,
  gamma_correction = TRUE,
  progress = interactive()
)
```

Arguments

image	Image filename or 3-layer RGB array.
depthmap	Depth map filename or 1d array.
focus	Defaults 0.5. Depth in which to blur. Minimum 0, maximum 1.
focallength	Default 100. Focal length of the virtual camera.
fstop	Default 4. F-stop of the virtual camera.
filename	Default NULL. The filename of the image to be saved. If this is not given, the image will be plotted instead.
preview	Default TRUE. If FALSE, it will not display the image and just return the RGB array.
preview_focus	Default FALSE. If TRUE, a red line will be drawn across the image showing where the camera will be focused.
bokehshape	Default circle. Also built-in: hex. The shape of the bokeh. If the user passes in a 2D matrix, that matrix will control the shape of the bokeh.
bokehintensity	Default 1. Intensity of the bokeh when the pixel intensity is greater than bokehlimit.
bokehlimit	Default 0.8. Limit after which the bokeh intensity is increased by bokehintensity.
rotation	Default 0. Number of degrees to rotate the hexagonal bokeh shape.
aberration	Default 0. Adds chromatic aberration to the image. Maximum of 1.
gamma_correction	Default TRUE. Controls gamma correction when adding colors. Default exponent of 2.2.
progress	Default TRUE. Whether to display a progress bar.

Value

3-layer RGB array of the processed image.

Examples

```
#Plot the dragon
plot_image(dragon)

#Plot the depth map
image(dragondepth, asp = 1, col = grDevices::heat.colors(256))

#Preview the focal plane:

render_bokeh(dragon, dragondepth, focus=950, preview_focus = TRUE)

#Change the focal length:

render_bokeh(dragon, dragondepth, focus=950, focallength=300)

#Add chromatic aberration:

render_bokeh(dragon, dragondepth, focus=950, focallength=300, aberration = 0.5)

#Change the focal distance:

render_bokeh(dragon, dragondepth, focus=600, focallength=300)
render_bokeh(dragon, dragondepth, focus=1300, focallength=300)

#Change the bokeh shape to a hexagon:

render_bokeh(dragon, dragondepth, bokehshape = "hex",
            focallength=300, focus=600)

#Change the bokeh intensity:

render_bokeh(dragon, dragondepth,
            focallength=400, focus=900, bokehintensity = 1)
render_bokeh(dragon, dragondepth,
            focallength=400, focus=900, bokehintensity = 3)

#Rotate the hexagonal shape:

render_bokeh(dragon, dragondepth, bokehshape = "hex", rotation=15,
            focallength=300, focus=600)
```

`render_convolution` *Render Convolution*

Description

Takes an image and applys a convolution operation to it, using a user-supplied or built-in kernel. Edges are calculated by limiting the size of the kernel to only that overlapping the actual image (renormalizing the kernel for the edges).

Usage

```
render_convolution(
  image,
  kernel = "gaussian",
  kernel_dim = 11,
  kernel_extent = 3,
  min_value = NULL,
  filename = NULL,
  preview = FALSE,
  gamma_correction = TRUE,
  progress = FALSE
)
```

Arguments

image	Image filename or 3-layer RGB array.
kernel	Default gaussian. By default, an 11x11 Gaussian kernel with a mean of 0 and a standard deviation of 1, running from -kernel_extent to kernel_extent. If numeric, this will be the standard deviation of the normal distribution. If a matrix, it will be used directly as the convolution kernel (but resized always to be an odd number of columns and rows).
kernel_dim	Default 11. The dimension of the gaussian kernel. Ignored if user specifies their own kernel.
kernel_extent	Default 3. Extent over which to calculate the kernel.
min_value	Default NULL. If numeric, specifies he minimum value (for any color channel) for a pixel to have the convolution performed.
filename	Default NULL. The filename of the image to be saved. If this is not given, the image will be plotted instead.
preview	Default TRUE. Whether to plot the convolved image, or just to return the values.
gamma_correction	Default TRUE. Controls gamma correction when adding colors. Default exponent of 2.2.
progress	Default TRUE. Whether to display a progress bar.

Value

3-layer RGB array of the processed image.

Examples

```
#Perform a convolution with the default gaussian kernel

plot_image(dragon)

#Perform a convolution with the default gaussian kernel
render_convolution(dragon, preview = TRUE)

#Increase the width of the kernel

render_convolution(dragon, kernel = 2, kernel_dim=21,kernel_extent=6, preview = TRUE)


#Only perform the convolution on bright pixels (bloom)

render_convolution(dragon, kernel = 5, kernel_dim=24, kernel_extent=24,
                   min_value=1, preview = TRUE)


#Use a built-in kernel:

render_convolution(dragon, kernel = generate_2d_exponential(falloff=2, dim=31, width=21),
                   preview = TRUE)


#We can also apply this function to matrices:

volcano %>% image()
volcano %>%
 render_convolution(kernel=generate_2d_gaussian(sd=1,dim=31)) %>%
 image()


#Use a custom kernel (in this case, an X shape):
custom = diag(10) + (diag(10)[,10:1])

plot_image(custom)
render_convolution(dragon, kernel = custom, preview = TRUE)
```

---

render_reorient *Reorient Image*

Description

Reorients an image or matrix. Transformations are applied in this order: x, y, and transpose.

Usage

```
render_reorient(
  image,
  flipx = FALSE,
  flipy = FALSE,
  transpose = FALSE,
  filename = NULL,
  preview = FALSE
)
```

Arguments

image	Image filename, 3-layer RGB array, or matrix.
flipx	Default FALSE. Flip horizontally
flipy	Default FALSE. Flip vertically.
transpose	Default FALSE. Transpose image.
filename	Default NULL. The filename of the image to be saved. If this is not given, the image will be plotted instead.
preview	Default FALSE. Whether to plot the convolved image, or just to return the values.

Value

3-layer RGB reoriented array or matrix.

Examples

```
#Original orientation

plot_image(dragon)


#Flip the dragon image horizontally

dragon %>%
 render_reorient(flipx = TRUE) %>%
 plot_image()


#Flip the dragon image vertically

dragon %>%
 render_reorient(flipy = TRUE) %>%
 plot_image()


#'#Transpose the dragon image

dragon %>%
 render_reorient(transpose = TRUE) %>%
```

```
 plot_image()
```

---

render_resized *Resize Image*

Description

Resizes an image or a matrix, using bilinear interpolation.

Usage

```
render_resized(image, mag = 1, dims = NULL, filename = NULL, preview = FALSE)
```

Arguments

image	Image filename, 3-layer RGB array, or matrix.
mag	Default 1. Amount to magnify the image, preserving aspect ratio. Overridden if dim is not NULL.
dims	Default NULL. Exact resized dimensions.
filename	Default NULL. The filename of the image to be saved. If this is not given, the image will be plotted instead.
preview	Default FALSE. Whether to plot the convolved image, or just to return the values.

Value

3-layer RGB resized array or matrix.

Examples

```
#Plot the image with a title

dragon %>%
 add_title("Dragon", title_offset=c(10,10), title_bar_color="black",
           title_size=20, title_color = "white") %>%
 plot_image()


#Half of the resolution

render_resized(dragon, mag = 1/2) %>%
 add_title("Dragon (half res)", title_offset=c(5,5), title_bar_color="black",
           title_size=10, title_color = "white") %>%
 plot_image()

#Double the resolution

render_resized(dragon, mag = 2) %>%
```

```
 add_title("Dragon (2x res)", title_offset=c(20,20), title_bar_color="black",
           title_size=40, title_color = "white") %>%
 plot_image()

#Specify the exact resulting dimensions

render_resized(dragon, dim = c(320,160)) %>%
 add_title("Dragon (custom size)", title_offset=c(10,10), title_bar_color="black",
           title_size=20, title_color = "white") %>%
 plot_image()
```

Index