Package 'MaxContrastProjection'

April 16, 2019

Type Package

Title Perform a maximum contrast projection of 3D images along the z-dimension into 2D

Version 1.6.1

Date 2017-02-08

Author Jan Sauer, Bernd Fischer

Maintainer Jan Sauer <jan.sauer@dkfz-heidelberg.de>

Description A problem when recording 3D fluorescent microscopy images is how to properly present these results in 2D. Maximum intensity projections are a popular method to determine the focal plane of each pixel in the image. The problem with this approach, however, is that out-of-focus elements will still be visible, making edges and fine structures difficult to detect. This package aims to resolve this problem by using the contrast around a given pixel to determine the focal plane, allowing for a much cleaner structure detection than would be otherwise possible. For convenience, this package also contains functions to perform various other types of projections, including a maximum intensity projection.

License Artistic-2.0

LazyLoad true

Imports EBImage, stats, methods

Depends R (>= 3.4)

VignetteBuilder knitr

Suggests knitr, BiocStyle, testthat

SystemRequirements GNU make

biocViews ImmunoOncology, CellBasedAssays, Preprocessing, Software, Visualization

RoxygenNote 6.0.1

git_url https://git.bioconductor.org/packages/MaxContrastProjection

git_branch RELEASE_3_8

git_last_commit 3dad069

git_last_commit_date 2019-01-04

Date/Publication 2019-04-15

R topics documented:

`calcContrast` *Calculate Local Image Contrast*

Description

Calculate the local contrast of an input image for every pixel of the image over a given window centered on that pixel.

Usage

```
calcContrast(image, w_x, w_y, brushShape = "disc", validate = TRUE)
```

Arguments

`image`	A numeric 2D matrix-like on which the contrast should be determined
`w_x`	The size of the window in x-direction
`w_y`	The size of the window in y-direction
`brushShape`	A string indicating the shape of the window. Currently supported values are `"box"`, `"disc"` and the default is "disc".
`validate`	A boolean value indicating if the variables need to be validated or if this function is being called internally, i.e. the variables have already been validated once. This is only used to marginally speed up internal calls to functions and has no bearing on the actual functionality of the method.

Details

The local contrast is calculated for every pixel of `image`. This means that a window is centered around a given pixel and the variance of the intensity values within this window are determined via Var = E[X^2] - (E[X])^2.

The `brushShape` indicates the shape of the window over which to calculate the variance. Depending on the symmetry of the objects being imaged, the window shape may have a significant impact on the quality of the projection.

Value

A 2D matrix with the same dimensions as `image`, which shows the local contrast at each corresponding pixel of `image`.

Author(s)

Jan Sauer

Examples

```
print(calcContrast)
```

---

cells *A sample segment of an organoid*

Description

A sample segment of an organoid

Author(s)

Jan Sauer

---

fixGaussianBlur *Remove Gaussian Blur*

Description

Fix the contrast projection error around very bright objects on a dark background

Usage

```
fixGaussianBlur(imageStack, indexMap, blur.size, validate = TRUE)
```

Arguments

`imageStack`	A numeric 3D array-like which should ne projected. The dimensions should be (spatial_1, spatial_2, numer_of_images)
`indexMap`	A custom index map according to which the image stack is projected. The values must be integers between 1 and the number of layers in `imageStack`
`blur.size`	An integer indicating the radius of the gaussian blur. The total diameter of the brush is defined as 2*blur.size+1, meaning that a blur.size' value of 0 will result in a 1x1 pixel brush.
`validate`	A boolean indicating if the function arguments should be validated. This is only used to marginally speed up internal calls to functions and has no bearing on the actual functionality of the method.

Details

Bright objects on dark backgrounds cause projection artefacts, a sort of gaussian "shadow" of the object. This is due to the unfocused images having a higher contrast in the regions directly outside of bright foreground objects than the actual background. This leads to ring shapes around the bright foreground which need to be removed. This is done by detecting bright objects with a primitive, intensity-based segmentation method (Otsu-thresholding) and determining a region around the foreground objects in which the gaussian blurring is visible. In this region, instead of using the determined z-indices for the projection, a Voronoi propagation starting from the closest foreground pixels is performed to ensure that the same z-layer is used for the nearby background as for the foreground.

Value

An index map with the corrected values around the foreground objects

Author(s)

Jan Sauer

Examples

```
print(fixGaussianBlur)
```

`getContrastStack` *Maximum contrast projection of a 3D image stack*

Description

Projects a z-stack of 2D images according to the highest local contrast. Optionally, median smoothing can be applied to the resulting projection index map prior to the projection itself.

Usage

```
getContrastStack(imageStack, w_x, w_y, brushShape = "disc", validate = TRUE)

getIndexMap(contrastStack, smoothing = 0, validate = TRUE)

contrastProjection(imageStack, w_x, w_y = NULL, smoothing = 0,
    brushShape = "disc", interpolation = 0, fix.gaussian.blur = FALSE,
    blur.size = 0, return.all = FALSE)

projection_fromMap(imageStack, indexMap, interpolation = 0, validate = TRUE)
```

Arguments

`imageStack`	A numeric 3D array-like which should ne projected. The dimensions should be (spatial_1, spatial_2, numer_of_images)
`w_x`	The size of the window in x-direction
`w_y`	The size of the window in y-direction
`brushShape`	A string indicating the shape of the window. Currently supported values are `"box"`, `"disc"` and "disc" is the default

validate	A boolean value indicating if the variables need to be validated or if this function is being called internally, i.e. the variables have already been validated once. This is only used to marginally speed up internal calls to functions and has no bearing on the actual functionality of the method.
contrastStack	A numeric 3D array-like which contains the local contrasts for each image in imageStack at each pixel
smoothing	The size of the median filter window. If this is 0, median smoothing is not applied.
interpolation	The size of the blurring kernel to use when interpolating values at the boundaries of regions in the index map. Set to 0 for no interpolation.
fix.gaussian.blur	A logical value indicating whether the false gaussian blur caused by unfocused images should be fixed or not (see vignette for more details on this).
blur.size	An integer indicating the radius of the gaussian blur. Ignored unless fix.gaussian.blur = TRUE. The total diameter of the brush is defined as 2*blur.size+1, meaning that a 'blur.size' value of 0 will result in a 1x1 pixel brush.
return.all	A logical value indicating whether only the projection should be returned (FALSE) or if all intermediate results should be returned as well, including the index map and the contrast stack (TRUE)
indexMap	A custom index map according to which the image stack is projected. The values must be integers between 1 and the number of layers in imageStack

Details

The local contrast for every image in the stack is determined using calcContrast. getContrastStack returns this stack of contrast maps. Then, the z-layer with the highest local contrast is determined for each pixel in the (x, y)-plane, resulting in an index map with the same spatial dimensions as the input images. This index map can then be smoothed with a median filter if desired. getIndexMap returns this index map. Lastly, the image stack is projected into the (x, y)-plane using this index map to determine which z-layer to use at every pixel. contrastProjection returns this fully projected image.

The brushShape indicates the shape of the window over which to calculate the variance. Depending on the symmetry of the objects being imaged, the window shape may have a significant impact on the quality of the projection.

If an object lies in several focal plains then the projection may include some artifical boundaries at the edges of the regions in each focal plain. Linear interpolation between the two layers at their boundaries serves to eliminate this problem. The interpolation size gives the size of the kernel to use for blurring the boundaries between individual regions of the index map. The projection values at these boundaries are then interpolated based on the non-integer values on the index maps. For example, if a pixel on the index map has the value 7.25, then the projected value at this pixel is 75

If a very bright object lies on a dark background, then the gaussian blurring of the unfocused image stacks can create a brighter ring structure around this object. Fixing this involves Voronoi propagation into the regions directly surrounding bright objects. This correction only makes sense if there is a clear differentiation between fore- and background in the image. A perfect segmentation is unnecessary as the rings will only appear around exceptionally bright objects, which are easy to segment.

Value

contrastProjection A 2D matrix corresponding to the maximum contrast projection of imageStack

getIndexMap A 2D matrix indicating the z-layer with the maximum contrast at every pixel in the $(x, y) - plane$ of `imageStack`

getContrastStack a 3D array corresponding to the contrast map for every image of `imageStack`

projection_fromMap A 2D matrix corresponding to the maximum contrast projection of `imageStack`

Functions

- `getContrastStack`: Get the full stack of contrast maps for each image in the image stack
- `getIndexMap`: Get the index map (with or without smoothing) which indicates the layer corresponding to the highest contrast for each pixel in the (x, y)-plane
- `projection_fromMap`: Get the index map (with or without smoothing) which indicates the layer corresponding to the highest contrast for each pixel in the (x, y)-plane

Author(s)

Jan Sauer

Examples

```
print(contrastProjection)
print(getIndexMap)
print(getContrastStack)
```

`intensityProjection` *Intensity Projection*

Description

Perform an intensity projection of a stack of images. The type of projection depends on the `projType`

Usage

```
intensityProjection(imageStack, projType = "max")
```

Arguments

`imageStack`	A numeric 3D array-like which should ne projected. The dimensions should be (spatial_1, spatial_2, numer_of_images)
`projType`	A string indicating the type of projection. Defaults to "max" and can take on the following values: `"max"`, `"min"`, `"mean"`, `"median"`, `"sd"`, `"sum"`

Details

The `projType` determines the time of projection to be used:

max Each pixel of the output image takes on the maximum value of the z-stack underneath the corresponding pixel of the input image stack.

min Each pixel of the output image takes on the minimum value of the z-stack underneath the corresponding pixel of the input image stack.

mean Each pixel of the output image takes on the mean value of the z-stack underneath the corresponding pixel of the input image stack.

median Each pixel of the output image takes on the median value of the z-stack underneath the corresponding pixel of the input image stack.

sd Each pixel of the output image takes on the standard deviation of the values of the z-stack underneath the corresponding pixel of the input image stack.

sum Each pixel of the output image takes on the sum of the values of the z-stack underneath the corresponding pixel of the input image stack.

Value

A 2D matrix corresponding to the maximum intensity projection of `imageStack`

Author(s)

Jan Sauer

Examples

```
dat = array(c(1,1,2,2,1,2,3,1), dim = c(2,2,2))
proj = intensityProjection(dat, projType = "max")
print(proj)
```

MaxContrastProjection *MaxContrastProjection: A package for performing z-projections of image stacks*

Description

The package MaxContrastProjection provides functions to perform the common intensity projections (max, min, etc.) as well as a maximum contrast projection we introduce here.

Examples

```
data(cells)
proj = contrastProjection(imageStack = cells, w_x = 15, smoothing = 5)
```

---

validateVariables *Validate Function Arguments*

Description

Validate the function arguments based on predefined rules. This function has no application for users and serves only to ensure correct variables. It either returns TRUE if all variables conform to the rules or stops execution in case one of the conditions is not met.

Usage

```
validateVariables(imageStack, image, w_x, w_y, smoothing, brushShape,
    projType, interpolation, contrastStack, indexMap, fix.gaussian.blur,
    blur.size, return.all)
```

Arguments

imageStack	A numeric 3D array-like
image	A 2D array-like
w_x	A numeric value
w_y	A numeric value
smoothing	An integer value
brushShape	A string
projType	A string
interpolation	An integer value
contrastStack	A numeric 3D array-like
indexMap	A numeric 2D array-like. Must always be validated simultaneously with imageStack
fix.gaussian.blur	A logical parameter, or one which can be coerced to a logical value
blur.size	An integer value
return.all	A logical parameter, or one which can be coerced to a logical value

Value

A boolean indicating that the function ran without error.

Author(s)

Jan Sauer

Examples

```
print(validateVariables)
```

Index