Package ‘flowcatchR’

May 21, 2024

Type Package

Title Tools to analyze in vivo microscopy imaging data focused on tracking flowing blood cells

Version 1.38.0

Date 2023-04-20

Description flowcatchR is a set of tools to analyze in vivo microscopy imaging data, focused on tracking flowing blood cells. It guides the steps from segmentation to calculation of features, filtering out particles not of interest, providing also a set of utilities to help checking the quality of the performed operations (e.g. how good the segmentation was). It allows investigating the issue of tracking flowing cells such as in blood vessels, to categorize the particles in flowing, rolling and adherent. This classification is applied in the study of phenomena such as hemostasis and study of thrombosis development. Moreover, flowcatchR presents an integrated workflow solution, based on the integration with a Shiny App and Jupyter notebooks, which is delivered alongside the package, and can enable fully reproducible bioimage analysis in the R environment.

License BSD_3_clause + file LICENSE

VignetteBuilder knitr

Suggests BiocStyle, knitr, rmarkdown

Depends R (>= 2.10), methods, EBImage

Imports colorRamps, abind, BiocParallel, graphics, stats, utils, plotly, shiny

SystemRequirements ImageMagick

LazyData true

URL https://github.com/federicomarini/flowcatchR, https://federicomarini.github.io/flowcatchR/

BugReports https://github.com/federicomarini/flowcatchR/issues

biocViews Software, Visualization, CellBiology, Classification, Infrastructure, GUI, ShinyApps

RoxygenNote 7.2.3

Encoding UTF-8

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

git_branch RELEASE_3_19

git_last_commit d07a179

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-21

Author Federico Marini [aut, cre] (<https://orcid.org/0000-0003-3252-7758>)

Maintainer Federico Marini <marinif@uni-mainz.de>

Contents

`add.contours` *Add object contours to a* `Frames` *object Creates a* `Frames` *object containing raw information, combined with the segmented images and the relative trajectory under analysis*

Description

If a `TrajectorySet` is provided and mode is set to `trajectories`, returns a `Frames` with all trajectories included in the IDs vector painted accordingly. If the mode is set to `particles`, it will just plot the particles (all) on all frames. If no `TrajectorySet` is provided, it will be computed with default parameters. If no `binary.frames` is provided, it will be computed also with default parameters

Usage

```
add.contours(
  raw.frames,
  binary.frames = NULL,
  trajectoryset = NULL,
  trajIDs = NULL,
  mode = "particles",
  col = NULL,
  channel = NULL
)
```

Arguments

`raw.frames`	A `Frames` object with raw images
`binary.frames`	A `Frames` object with preprocessed frames
`trajectoryset`	A `TrajectorySet` object

trajIDs	Numeric vector, the ID(s) of the trajectory.
mode	A character string, can assume the values particles or trajectories. Defaults to particles
col	A vector of color strings
channel	A character string, to select which channel to process

Value

A new Frames object with contours of the objects added

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("MesenteriumSubset")
## Not run:
paintedTrajectories <- add.contours(raw.frames = MesenteriumSubset,
                                    mode = "trajectories",channel="red")
paintedParticles <- add.contours(raw.frames = MesenteriumSubset,
                                 mode = "particles",channel="red")
inspect.Frames(paintedTrajectories)
inspect.Frames(paintedParticles)

## End(Not run)
```

---

addParticles *Combines the information from a raw* Frames *object and the corresponding preprocessed one*

Description

All objects are painted with a unique colour - for sake of speed

Usage

```
addParticles(raw.frames, binary.frames, col = NULL)
```

Arguments

raw.frames	A Frames object containing the raw images
binary.frames	A Frames object with the preprocessed versions of the images (e.g. segmented)
col	A color character string, to select which color will be used for drawing the contours of the particles. If not specified, it will default according to the objects provided

Value

A `Frames` object, whose images are the combination of the raw images with the segmented objects drawn on them

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

---

axesInfo *Info on the dimensions of the FOV*

Description

Auxiliary function to return the dimensions of the field of interest

Usage

```
axesInfo(frames)
```

Arguments

`frames`	A `Frames` object

Value

A list object, containing the extremes of the field of interest (x-y-z, where z is time)

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

---

candidate.platelets *A sample* `ParticleSet` *object*

Description

The sample `ParticleSet` object is constituted by the platelets identified from the `MesenteriumSubset` data

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

channel.Frames *Channel extraction for* objects

Description

channel

Usage

```
channel.Frames(frames, mode)
```

Arguments

frames	A Frames object
mode	A character value specifying the target mode for conversion.

Value

A Frames object with just the infotmation on the selected channel

Examples

```
data("MesenteriumSubset")
channel.Frames(MesenteriumSubset,"red")
```

computeMSD *Calculates the Mean Squared Displacement for a trajectory*

Description

Calculates the Mean Squared Displacement for a trajectory

Usage

```
computeMSD(sx, sy, until = 4)
```

Arguments

sx	x axis positions along the trajectory
sy	y axis positions along the trajectory
until	how many points should be included in the Mean Squared Displacement curve

Value

A numeric vector containing the values of the MSD

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

---

crop.Frames *Cut borders of a* Frames *object*

Description

Performs cropping on the Frames object, selecting how many pixels should be cut on each side

Usage

```
crop.Frames(
  frames,
  cutLeft = 5,
  cutRight = 5,
  cutUp = 5,
  cutDown = 5,
  cutAll = 0,
  testing = FALSE,
  ...
)
```

Arguments

frames	An input Frames object
cutLeft	Amount of pixels to be cut at the side
cutRight	Amount of pixels to be cut at the side
cutUp	Amount of pixels to be cut at the side
cutDown	Amount of pixels to be cut at the side
cutAll	Amount of pixels to be cut at all sides. Overrides the single side values
testing	Logical, whether to just test the cropping or to actually perform it. Default set to FALSE
...	Arguments to be passed to display (e.g. setting the method argument)

Details

Cropping can be performed with careful choice of all cutting sides, or cropping a single value from all sides

Value

A Frames object, with cropped frames in the image slot

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("MesenteriumSubset")
crop.Frames(MesenteriumSubset)
```

---

export.Frames *Exports a* Frames *object*

Description

Writes the images contained in the image slot of the Frames object elements. The images can be exported as single frames, or as a .gif image that is composed by the single frames.

Usage

```
export.Frames(
  frames,
  dir = tempdir(),
  nameStub = "testExport",
  createGif = FALSE,
  removeAfterCreatingGif = TRUE
)
```

Arguments

frames	A Frames object
dir	The path of the folder where the image should be written
nameStub	The stub for the file name, that will be used as a prefix for the exported images
createGif	Logical, whether to create or not an animated .gif file
removeAfterCreatingGif	Logical, whether to remove the single exported .png images after creating the single .gif

Value

Image files are written in the desired location

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("MesenteriumSubset")
## Not run: export.Frames(MesenteriumSubset,nameStub="subset_export_",
                        createGif=TRUE,removeAfterCreatingGif=FALSE)
## End(Not run)
```

---

export.particles *Exports a* ParticleSet *object*

Description

Writes the particles contained in the particles data frame slot of the ParticleSet object elements. A track of the provenience of the particles is stored as a comment line above the header

Usage

```
export.particles(
  particleset,
  dir = tempdir(),
  nameStub = "testExport_particles"
)
```

Arguments

particleset	A ParticleSet object
dir	The path of the folder where the particle sets should be written
nameStub	The stub for the file name, that will be used as a prefix for the exported particle sets

Value

Particle sets files are written in the desired location

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("candidate.platelets")
## Not run: export.particles(candidate.platelets)
```

extractKinematics.traj

Calculate a set of kinematics parameters from a single trajectory

Description

The computed set of parameters include `delta.x`, `delta.t` and `delta.v` (displacements and instantaneous velocity), `totalTime`, `totalDistance`, `distStartToEnd`, `curvilinearVelocity`, `straightLineVelocity` and `linearityForwardProgression`, Mean Squared Displacement, velocity autocorrelation, and more

Usage

```
extractKinematics.traj(
  trajectoryset,
  trajectoryID,
  acquisitionFrequency = 30,
  scala = 50
)
```

Arguments

`trajectoryset`	A `TrajectorySet` object
`trajectoryID`	The ID of a single trajectory
`acquisitionFrequency`	The frame rate of acquisition for the images, in milliseconds
`scala`	The value of micro(?)meters to which each single pixel corresponds

Value

A `KinematicsFeatures` object

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

flowcatchR-pkg

flowcatchR: analyzing time-lapse microscopy imaging, from detection to tracking

Description

A toolset to analyze in vivo microscopy imaging data focused on tracking flowing blood cells.

Details

flowcatchR is a set of tools to analyze in vivo microscopy imaging data, focused on tracking flowing blood cells. It guides the steps from segmentation to calculation of features, filtering out particles not of interest, providing also a set of utilities to help checking the quality of the performed operations (e.g. how good the segmentation was). The main novel contribution investigates the issue of tracking flowing cells such as in blood vessels, to categorize the particles in flowing, rolling and adherent. This classification is applied in the study of phenomena such as hemostasis and study of thrombosis development.

Author(s)

Federico Marini <marinif@uni-mainz.de>, Johanna Mazur <mazur@uni-mainz.de>, Harald Binder <binderh@uni-mainz.de>, 2015

Maintainer: Federico Marini <marinif@uni-mainz.de>

Frames *Constructor for a* Frames *object*

Description

Constructor for a Frames object

Usage

```
Frames(x, channel)
```

Arguments

x	A multi-dimensional Image object
channel	A character vector, can be 'red','green','blue' or 'all' (if in color mode)

Value

The created Frames object.

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("MesenteriumSubset")
inputImg <- Image(MesenteriumSubset)
Frames(inputImg,"red")
```

Frames-class *Frames class*

Description

S4 class for storing information on multiple images belonging to the same time-lapse experiment. It is designed as a subclass of the existing Image class from the EBImage package

Slots

`channel` A character vector, can be 'red','green','blue' or 'all' (if in color mode)

initialize.LinkedParticleSet *Initialize a* `ParticleSet` *object for subsequent linking/tracking*

Description

Initialize a `ParticleSet` object for subsequent linking/tracking

Usage

```
initialize.LinkedParticleSet(particleset, linkrange = 1)
```

Arguments

`particleset`	A `ParticleSet` object
`linkrange`	The number of frames to look for candidate particles potentially belonging to the same track

Value

A `ParticleSet` object with slots dedicated for the tracking pre-filled

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

`inspect.Frames` *Explore the frames of a* `Frames`

Description

The first frames of a `Frames` are displayed in the browser, and are interactively navigable.

Usage

```
inspect.Frames(
  frames,
  nframes = NULL,
  display.method = "browser",
  verbose = FALSE
)
```

Arguments

`frames`	A `Frames` object
`nframes`	The number of frames to display (default value: `NULL`, all are displayed)
`display.method`	Method for displaying, can be either `raster` or `browser`. Defaults to `browser`, by opening a window in the browser
`verbose`	Logical, whether to provide additional output on the command line alongside with the images themselves

Value

`inspect.Frames` returns an invisible `NULL`.

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("MesenteriumSubset")
## Not run: inspect.Frames(MesenteriumSubset)
```

`kinematics` *Calculate a set of kinematics parameter from a* `TrajectorySet` *object, or a single parameter, or from a single trajectory (all possible combinations)*

Description

The computed set of parameters include `delta.x`, `delta.t` and `delta.v` (displacements and instantaneous velocity), `totalTime`, `totalDistance`, `distStartToEnd`, `curvilinearVelocity`, `straightLineVelocity` and `linearityForwardProgression`, Mean Squared Displacement, velocity autocorrelation, and more. If a single trajectory is specified, the computation is performed for that trajectory alone. If a parameter is specified, only that parameter is reported, either for one or all trajectories

Usage

```
kinematics(
  trajectoryset,
  trajectoryIDs = NULL,
  acquisitionFrequency = 30,
  scala = 50,
  feature = NULL
)
```

Arguments

`trajectoryset`	A `TrajectorySet` object
`trajectoryIDs`	The ID of a single trajectory
`acquisitionFrequency`	The frame rate of acquisition for the images, in milliseconds
`scala`	The value of micro(?)meters to which each single pixel corresponds
`feature`	Character string, the name of the feature to be computed

Value

A `KinematicsFeaturesSet` object, or a `KinematicsFeatures` object, or an atomic value, or a list(eventually coerced to a vector)

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("candidate.platelets")
platelets.trajectories <- trajectories(candidate.platelets)
# for all trajectories, all features
alltrajs.features <- kinematics(platelets.trajectories)
```

```
# for one trajectory, all features
traj11features <- kinematics(platelets.trajectories,trajectoryIDs = 11)
# for all trajectories, one feature
alltrajs.curvVel <- kinematics(platelets.trajectories,feature = "curvilinearVelocity")
```

`KinematicsFeatures-class` *KinematicsFeatures class*

Description

S4 class for storing information on all kinematics features identified for a single trajectory

Slots

`.Data` A list storing the information for the kinematics features

---

`KinematicsFeaturesSet-class` *KinematicsFeaturesSet class*

Description

S4 class for storing information on all kinematics features identified for all trajectories. Single `KinematicsFeatures` objects are the element of the main list

Slots

`.Data` A list storing the information for the sets of kinematics features

---

`length.Frames` *Compute the length of render frames in a* `Frames` *object*

Description

Compute the length of render frames in a `Frames` object

Usage

```
## S3 method for class 'Frames'
length(x)
```

Arguments

x	A Frames object

Value

An integer number

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("MesenteriumSubset")
length(MesenteriumSubset)
```

---

link.particles *Links a* ParticleSet *object*

Description

Performs linking of the particles by tracking them through the frames

Usage

```
link.particles(
  particleset,
  L,
  R = 2,
  epsilon1 = 0.1,
  epsilon2 = 2,
  lambda1 = 1,
  lambda2 = 1,
  penaltyFunction = penaltyFunctionGenerator(),
  verboseOutput = FALSE,
  prog = FALSE,
  include.intensity = TRUE,
  include.area = FALSE
)
```

Arguments

particleset	A ParticleSet object
L	Maximum number of pixels an object can move in two consecutive frames
R	Linkrange, i.e. the number of consecutive frames to search for potential candidate links

`epsilon1` A numeric value, to be used in the formula. Jitter for allowing angular displacements

`epsilon2` A numeric value, to be used in the formula. Jitter for allowing spatial displacements

`lambda1` A numeric value. Multiplicative factor for the penalty function

`lambda2` A numeric value. Multiplicative factor applied to the angular displacement

`penaltyFunction` A function structured in such a way to be applied as penalty function in the linking

`verboseOutput` Logical, whether the output should report additional intermediate steps. For debugging use mainly

`prog` Logical, whether the a progress bar should be shown during the tracking phase

`include.intensity` Logical, whether to include also intensity change of the particles in the cost function calculation

`include.area` Logical, whether to include also area change of the particles in the cost function calculation

Value

A `LinkedParticleSet` object

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

References

I F Sbalzarini and P Koumoutsakos."Feature point tracking and trajectory analysis for video imaging in cell biology." In: Journal of structural biology 151.2 (Aug. 2005), pp. 182-95. ISSN: 1047-8477. DOI: 10.1016/j.jsb.2005.06.002. URL: http://www.ncbi.nlm.nih.gov/pubmed/16043363

Examples

```
data("candidate.platelets")
tracked.platelets <- link.particles(candidate.platelets, L= 40)
```

LinkedParticleSet-class *LinkedParticleSet class*

Description

S4 class for storing information of particles after they have been tracked. It inherits the slots from the `ParticleSet` class.

Slots

`tracking` A list storing all necessary information for the tracking algorithm to work, and for providing the information to the function to determine the trajectories

matchTrajToParticles *Match trajectories to related particles.*

Description

Match trajectories to the related particles in the `TrajectorySet` and `ParticleSet` objects. This function returns a new `ParticleSet` object that contains as additional column the trajectory ID that the particular particle was assigned to. Used also by other routines, such as `snap`

Usage

```
matchTrajToParticles(particleset, trajectoryset)
```

Arguments

`particleset`	A `ParticleSet` object
`trajectoryset`	A `TrajectorySet` object coupled to the `particleset`

Value

A `ParticleSet` object with an additional column with the trajectory IDs

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2015

Examples

```
data(candidate.platelets)
trajs <- trajectories(candidate.platelets)
matchTrajToParticles(candidate.platelets, trajs)
```

MesenteriumSubset *A sample* Frames *object*

Description

The sample Frames object is constituted by a subset of a time-lapse intravital microscopy imaging dataset. Green channel marks leukocytes, red channel focuses on blood platelets. 20 frames are provided in this subset. Images are kindly provided by Sven Jaeckel (<Sven.Jaeckel@unimedizin-mainz.de>).

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

normalizeFrames *Normalize the values of a* Frames *object*

Description

Applies a transformation to the Frames object in a way that the intensities throughout the acquisition are normalized overall in term of pixel values sums. It can be used to compensate for example a global change in the illumination values, e.g. due to changed acquisition conditions in experiments that span long timescales.

Usage

```
normalizeFrames(frames, normFun = "median")
```

Arguments

frames	A Frames object to normalize
normFun	The normalization function chosen. Can be one of mean or median

Value

A Frames object with normalized pixel values.

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data(MesenteriumSubset)
normalizeFrames(MesenteriumSubset,normFun="median")
```

particles *Extracts particles from the images of a* `Frames` *object.*

Description

Extracts particles from the images of a `Frames` object.

Usage

```
particles(
  raw.frames,
  binary.frames = NULL,
  channel = NULL,
  BPPARAM = bpparam()
)
```

Arguments

`raw.frames`	A `Frames` object with the raw images (mandatory)
`binary.frames`	A `Frames` object with preprocessed images (optional, if not provided gets produced with standard default parameters)
`channel`	Character string. The channel to perform the operations on. Can be `red`, `green` or `blue`
`BPPARAM`	a `MulticoreParam` object, used to control the performances inside the `BiocParallel` call to process frames in parallel by taking advantage of the computing infrastructure available

Value

A `ParticleSet` object, containing all detected particles for each frame

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2015

Examples

```
data("MesenteriumSubset")
```

ParticleSet-class *ParticleSet class*

Description

S4 class for storing information on particles detected in distinct frames.

Slots

`.Data` A list storing the information for the particles

`channel` A character vector, can be 'red','green', or 'blue'. It refers to which channel the particles were detected

penaltyFunctionGenerator *Generate a penalty function*

Description

A function to generate penalty functions to use while linking particles

Usage

```
penaltyFunctionGenerator(
  epsilon1 = 0.1,
  epsilon2 = 2,
  lambda1 = 1,
  lambda2 = 1
)
```

Arguments

`epsilon1`	A numeric value, to be used in the formula. Jitter for allowing angular displacements
`epsilon2`	A numeric value, to be used in the formula. Jitter for allowing spatial displacements
`lambda1`	A numeric value. Multiplicative factor for the penalty function
`lambda2`	A numeric value. Multiplicative factor applied to the angular displacement

Value

A function object, to be used as penalty function

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
custom.function <- penaltyFunctionGenerator(epsilon1=0.1,epsilon2=6,lambda1=1.5,lambda2=0)
```

---

plot.TrajectorySet *3D representation of a* TrajectorySet *object*

Description

Provides a visual representation of a TrajectorySet object

Usage

```
## S3 method for class 'TrajectorySet'
plot(x, frames, verbose = FALSE, ...)
```

Arguments

x	A TrajectorySet object
frames	A Frames object, used here to identify the limits of the region of interest
verbose	Logical, whether to provide additional output on the command line
...	Arguments to be passed to methods

Details

Based on the plotly library, the function extracts the region of interests from the dimensions of an image of the Frames object, and afterwards plots the x-y-time representation of the identified trajectories

Value

plot.TrajectorySet returns an invisible NULL.

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("MesenteriumSubset")
data("candidate.platelets")
platelets.trajectories <- trajectories(candidate.platelets)
## Not run:
plot(platelets.trajectories,MesenteriumSubset)

## End(Not run)
```

---

plot2D.TrajectorySet *2D projection of a* TrajectorySet *object*

Description

Provides a bird's eye view of a TrajectorySet object on a bidimensional space

Usage

```
plot2D.TrajectorySet(
  trajectoryset,
  frames,
  trajIDs = NULL,
  addGrid = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

trajectoryset	A TrajectorySet object
frames	A Frames object, used here to identify the limits of the region of interest
trajIDs	A vector containing the ids of the desired trajectories
addGrid	Logical, add an additional grid to the 2-dimensional plot (visual aid for back-tracking trajectory point locations)
verbose	Logical, whether to provide additional output on the command line
...	Arguments to be passed to methods

Details

This function extracts the region of interests from the dimensions of an image of the Frames object, and afterwards plots the x-y-time representation of the identified trajectories on a 2d plane. It is possible to subset the TrajectorySet object with the IDs of the desired trajectories

Value

plot2D.TrajectorySet returns an invisible NULL.

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("MesenteriumSubset")
data("candidate.platelets")
platelets.trajectories <- trajectories(candidate.platelets)
plot2D.TrajectorySet(platelets.trajectories,MesenteriumSubset)
```

---

preprocess.Frames *Preprocessing function for* Frames *objects*

Description

Frames objects are processed according to the chosen set of parameters. Many of them refer directly to existing EBImage functions, please see the corresponding help for additional information

Usage

```
preprocess.Frames(
  frames,
  brush.size = 3,
  brush.shape = "disc",
  at.offset = 0.15,
  at.wwidth = 10,
  at.wheight = 10,
  kern.size = 3,
  kern.shape = "disc",
  ws.tolerance = 1,
  ws.radius = 1,
  displayprocessing = FALSE,
  ...
)
```

Arguments

frames	A Frames object
brush.size	Size in pixels of the brush to be used for initial smoothing (low-pass filtering)
brush.shape	Shape of the brush to be used for initial smoothing (low-pass filtering)
at.offset	Offset to be used in the adaptive thresholding step - see also thresh. As an alternative thresholding method, see also otsu in the EBImage package.
at.wwidth	Width of the window for the adaptive thresholding step - see also thresh. As an alternative thresholding method, see also otsu in the EBImage package.

at.wheight	Height of the window for the adaptive thresholding step - see also thresh. As an alternative thresholding method, see also otsu in the EBImage package.
kern.size	Size in pixels of the kernel used for morphological operations - e.g., opening, which is an erosion followed by a dilation, and closing which is a dilation followed by an erosion - see also opening, closing
kern.shape	Shape of the kernel used for morphological operations
ws.tolerance	Tolerance allowed in performing the watershed-based segmentation (see also watershed)
ws.radius	Radius for the watershed-based segmentation (see also watershed)
displayprocessing	Logical, whether to display intermediate steps while performing preprocessing. Dismissed currently, it could increase runtime a lot
...	Arguments to be passed to methods

Value

A Frames object, whose frame images are the preprocessed versions of the input images

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("MesenteriumSubset")
preprocess.Frames(channel.Frames(MesenteriumSubset,"red"))
```

---

read.Frames *Constructor for a* Frames *object*

Description

This function is used to create a Frames object from a vector of image files (or a folder specifying the directory containing them). The number of frames is also specified, as just a subset of the images can be used for this

Usage

```
read.Frames(image.files, nframes = NULL)
```

Arguments

image.files	Vector of strings containing the locations where the (raw) images are to be found, or alternatively, the path to the folder
nframes	Number of frames that will constitute the Frames object

Value

An object of the `Frames` class, which holds the info on a list of frames, specifying for each the following elements:

`image`	The `Image` object containing the image itself
`location`	The complete path to the location of the original image

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
## see vignette
## Not run: fullData <- read.Frames(image.files = "/path/to/the/directory", nframes = 100)
```

`read.particles` *Constructor for a* `ParticleSet` *object*

Description

This function is used to create a `ParticleSet` object from a vector/list of tab separated text files, each of one containing one line for each particle in the related frame, alongside with its coordinates and if available, the computed features The number of frames is also specified, as just a subset of the particle lists can be used for this

Usage

```
read.particles(particle.files, nframes = NULL)
```

Arguments

`particle.files`	Vector of strings containing the locations where the particle coordinates are to be found, or alternatively, the path to the folder
`nframes`	Number of frames that will constitute the `ParticleSet` object

Value

An object of the `ParticleSet` class

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
## see vignette and export.particles
```

repmat — *Function equivalent for MATLAB's repmat - Replicate and tile arrays*

Description

A more flexible and stylish alternative to replicate the behaviour of the repmat function of MATLAB

Usage

```
repmat(a, n, m)
```

Arguments

a	The matrix to copy
n	The n value for the tiling
m	The m value for the tiling

Value

Creates a large matrix consisting of an m-by-n tiling of copies of a.

Author(s)

Robin Hankin, 2001

References

http://cran.r-project.org/doc/contrib/R-and-octave.txt

rotate.Frames — *Rotates all images in a* `Frames` *object*

Description

Rotation is performed exploiting the rotate function of the `EBImage` package. Could be automated if support for coordinate/pixel interaction is included

Usage

```
rotate.Frames(frames, angle, testing = FALSE)
```

Arguments

frames	A `Frames` object
angle	The rotation angle (clockwise) specified in degrees
testing	Logical, whether to just test the rotation or to actually perform it. Default set to `FALSE`

Value

A `Frames` object containing the rotated frames

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("MesenteriumSubset")
rotate.Frames(MesenteriumSubset,angle = 40)
```

---

select.Frames — *Extracts subsets of frames from a* `Frames` *object*

Description

An input `Frames` object is subject to subsetting. This function is useful e.g. when the trajectory of interest is presenting gaps (i.e. does not actually include a frame)

Usage

```
select.Frames(frames, framesToKeep = 1, ...)
```

Arguments

`frames`	A `Frames` object
`framesToKeep`	A vector containing the indexes of the frames to keep in the selection
`...`	Arguments to be passed to methods

Value

A `Frames` object, composed by the subset of frames of the input `Frames`

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("MesenteriumSubset")
select.Frames(MesenteriumSubset, framesToKeep = c(1:10, 14:20))
```

`select.particles` *Performs filtering on a* `ParticleSet` *object*

Description

According to parameters of interests, such as size, eccentricity/shape, filters out the particles that do not satisfy the indicated requirements

Usage

```
select.particles(particleset, min.area = 1, max.area = 1000)
```

Arguments

`particleset`	A `ParticleSet` object. A `LinkedParticleSet` object can also be provided as input, yet the returned object will be a `ParticleSet` object that needs to be linked again
`min.area`	Size in pixels of the minimum area needed to detect the object as a potential particle of interest
`max.area`	Size in pixels of the maximum area allowed to detect the object as a potential particle of interest

Value

A `ParticleSet` object

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("candidate.platelets")
selected.platelets <- select.particles(candidate.platelets, min.area = 5)
selected.platelets
```

shinyFlow — *Shiny application for exploring the features and parameters provided by* flowcatchR

Description

Launches a Shiny Web Application for interactive data exploration. Default data loaded are the frames from the MesenteriumSubset object, custom values can be inserted by typing the location of the data stored in a local folder. The Application is structured in a variety of tabs that mirror the steps in the usual workflow in time-lapse microscopy images. These can allow the user to interactively explore the parameters and their effect in the reactive framework provided by Shiny.

Usage

```
shinyFlow()
```

Value

The Shiny Application is launched in the web browser

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2015

Examples

```
## Not run: shinyFlow()
```

show,Frames-method — *Display conveniently a* Frames *object*

Description

Display conveniently a Frames object

Usage

```
## S4 method for signature 'Frames'
show(object)
```

Arguments

object	A Frames object

Value

This returns an invisible NULL.

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("MesenteriumSubset")
print(MesenteriumSubset)
```

---

show,KinematicsFeatures-method

Displaying conveniently a `KinematicsFeatures` *object*

Description

Displaying conveniently a `KinematicsFeatures` object

Usage

```
## S4 method for signature 'KinematicsFeatures'
show(object)
```

Arguments

`object`	A `KinematicsFeatures` object

Value

This returns an invisible `NULL`.

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("candidate.platelets")
platelets.trajectories <- trajectories(candidate.platelets)
traj11features <- kinematics(platelets.trajectories,trajectoryIDs = 11)
print(traj11features)
```

show,KinematicsFeaturesSet-method

Display conveniently a `KinematicsFeatureSet` *object*

Description

Display conveniently a `KinematicsFeatureSet` object

Usage

```
## S4 method for signature 'KinematicsFeaturesSet'
show(object)
```

Arguments

`object`	A `KinematicsFeatureSet` object

Value

This returns an invisible `NULL`.

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("candidate.platelets")
platelets.trajectories <- trajectories(candidate.platelets)
alltrajs.features <- kinematics(platelets.trajectories)
print(alltrajs.features)
```

show,LinkedParticleSet-method

Display conveniently a `LinkedParticleSet` *object*

Description

Display conveniently a `LinkedParticleSet` object

Usage

```
## S4 method for signature 'LinkedParticleSet'
show(object)
```

Arguments

object	A LinkedParticleSet object

Value

This returns an invisible NULL.

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("candidate.platelets")
linked.platelets <- link.particles(candidate.platelets,L=26,R=3,epsilon1=0,
epsilon2=0,lambda1=1,lambda2=0,penaltyFunction=penaltyFunctionGenerator(),
include.area=FALSE)
print(linked.platelets)
```

---

show,ParticleSet-method

Display conveniently a ParticleSet *object*

Description

Display conveniently a ParticleSet object

Usage

```
## S4 method for signature 'ParticleSet'
show(object)
```

Arguments

object	A ParticleSet object

Value

This returns an invisible NULL.

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("candidate.platelets")
print(candidate.platelets)
```

show,TrajectorySet-method

Display conveniently a `TrajectorySet` *object*

Description

Display conveniently a `TrajectorySet` object

Usage

```
## S4 method for signature 'TrajectorySet'
show(object)
```

Arguments

`object`	A `TrajectorySet` object

Value

This returns an invisible `NULL`.

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("candidate.platelets")
platelets.trajectories <- trajectories(candidate.platelets)
print(platelets.trajectories)
```

snap

Snap the features of the closest particle identified

Description

This function combines all classes related to a single experiment in order to deliver a clickable feedback on one of the frames.

Usage

```
snap(
  raw.frames,
  binary.frames,
  particleset,
  trajectoryset,
  frameID = 1,
  infocol = "yellow",
  infocex = 1,
  showVelocity = FALSE
)
```

Arguments

raw.frames	A Frames object with the raw frames data
binary.frames	A Frames object with the preprocessed frames data
particleset	A ParticleSet object with the particles data
trajectoryset	A TrajectorySet object with the trajectories data
frameID	The ID of the frame to inspect
infocol	The color to use for plotting the contours and the information on the clicked particle
infocex	The numeric character expansion value as in cex to be used for printing the text on the image
showVelocity	Logical, whether to display additional information on the instantaneous velocity of the particle

Value

An image of the selected frame, rendered in R native graphics, and additionally a list with the coordinates as well as the trajectory ID of the particle closest to the clicked location

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2015

Examples

```
## Not run: data(MesenteriumSubset)
binary.frames <- preprocess.Frames(channel.Frames(MesenteriumSubset,"red"))
particleset <- particles(MesenteriumSubset,binary.frames,"red")
trajectoryset <- trajectories(particleset)
snap(MesenteriumSubset,binary.frames,particleset,trajectoryset,frameID=1)

## End(Not run)
```

toCartesianCoords *Converts polar coordinates to cartesian coordinates*

Description

Conversion from (radius,theta) to (x,y)

Usage

```
toCartesianCoords(Theta, Radius)
```

Arguments

Theta	The Theta angle
Radius	The radius value in polar coordinates

Value

A list containing Theta and Radius, as in polar coordinates

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

toPolarCoords *Converts cartesian coordinates to polar coordinates*

Description

Conversion from (x,y) to (radius,theta)

Usage

```
toPolarCoords(x, y)
```

Arguments

x	x coordinate
y	y coordinate

Value

A list containing Theta and Radius, as in polar coordinates

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

trajectories *Generate trajectories*

Description

Generates a `TrajectorySet` object from a (`Linked`)`ParticleSet`

Usage

```
trajectories(particleset, verbose = FALSE, ...)
```

Arguments

`particleset`	A (`Linked`)`ParticleSet` object
`verbose`	Logical, currently not used - could be introduced for providing additional info on the trajectories
`...`	Arguments to be passed to methods

Value

A `TrajectorySet` object

Author(s)

Federico Marini, <marinif@uni-mainz.de>, 2014

Examples

```
data("candidate.platelets")
platelets.trajectories <- trajectories(candidate.platelets)
```

TrajectorySet-class *TrajectorySet class*

Description

S4 class for storing information on the trajectories identified, including whether there were gaps, the number of points, and more

Slots

`.Data` A list storing the information for the particles

`channel` A character vector, can be 'red','green', or 'blue'. It refers to which channel the particles were detected

Index