# Package: bemovi.LEEF (via r-universe)

September 13, 2024

**Title** BEMOVI, software for extracting BEhaviour and MOrphology from VIdeos. This version is adapted for LEEF-UZH

**Description** An R and ImageJ based work flow to automatically measure behaviour and morphology from videos. Moving individuals are identified by background subtraction, morphology extracted, and trajectories assembled through time from coordinate data. Abundance, morphology and behaviour can be summarized based on trajectory data.

**Version** 1.1.0

**Author** Frank Pennekamp <Frank.Pennekamp@ieu.uzh.ch>, Owen Petchey <Owen.Petchey@ieu.uzh.ch>

**Maintainer** Rainer M Krug <Rainer@krugs.de>

**Imports** data.table, circular, settings, yaml, grDevices, graphics, parallel, loggit

**Depends** R (>= 3.0.2)

**Encoding** UTF-8



**LazyData** true

**RoxygenNote** 7.2.3

**Repository** https://leef-uzh.r-universe.dev

**RemoteUrl** https://github.com/LEEF-UZH/bemovi

**RemoteRef** LEEF

**RemoteSha** a20598c2df2ccfe3f3221a1bbdb76373e5dcfd07

## Contents

---

`anglefun` *Title*

---

## Description

Title

## Usage

```
anglefun(xx, yy, bearing = TRUE, as.deg = FALSE)
```

## Arguments

| | |
|---|---|
| `xx` | TODO |
| `yy` | TODO |
| `bearing` | TODO |
| `as.deg` | TODO |

## Value

TODO

`bemovi` *Bemovi, software for extracting BEhaviour and MOrphology from VIdeos.*

## Description

bemovi provides functions to segment videos of organisms, identify individuals by tracking their paths through time and measure their traits (morphology and movement) as well as their abundances

`calculate_mvt` *A function to calculate movement metrics for each trajectory*

## Description

The function takes the X- and Y-coordinates for each unqiue trajectory and calculates movement metrics such as the gross and net displacement, absolute and relative angles and duration

## Usage

```
calculate_mvt(
  data,
  to.data = par_to.data(),
  trajectory.data.folder = par_trajectory.data.folder(),
  pixel_to_scale = par_pixel_to_scale(),
  fps = par_fps()
)
```

## Arguments

| | |
|---|---|
| `data` | dataframe containing the X- and Y-coordinates, the frame and the trajectory ID |
| `to.data` | path to the working directory |
| `trajectory.data.folder` | directory where the data is saved |
| `pixel_to_scale` | specify how a pixel scales to real dimensions |
| `fps` | specify the frame rate of the video |

## Value

returns a data.table with the movement metrics for each fix appended to the original data (NB: movement metrics often need two (e.g. step length), sometimes even three (e.g., turning angles) fixes; fixes for which metrics cannot be calculated are padded with NA). The movement parameters are the step length, the step duration, the step speed (step length/step duration), the gross displacement as the cumulative sum of the step lengths, the net displacement between the first fix of a given trajectory and the current fix and finally the relative angle (turning angle) and absolute

angle (in radians). For details on these metrics, please refer to a dedicated textbook (e.g. Turch (1998): Quantitative Analysis of Movement: Measuring and Modeling Population Redistribution in Animals and Plants, Sinauer Associates, Sunderland).

---

`check_threshold_values`

*Function to assist finding good thresholds used for the segmentation of the difference video. If you run bemovi for the first time, verify with this function that all target objects are properly identified This function creates an ImageJ macro that can be helpful for checking the thresholds specified in the user section; the macro will be saved in the ImageJ macro directory in the working directory and then automatically opened in ImageJ. Depending on the video size, it might take some time to open the video and the thresholding menu. The default thresholds values should be adjusted in the ImageJ macro, until appropriate values are found. These values should then be used in the R functions / script, e.g., when calling the Locate_and_measure_particles function.*

---

## Description

Function to assist finding good thresholds used for the segmentation of the difference video. If you run bemovi for the first time, verify with this function that all target objects are properly identified

This function creates an ImageJ macro that can be helpful for checking the thresholds specified in the user section; the macro will be saved in the ImageJ macro directory in the working directory and then automatically opened in ImageJ. Depending on the video size, it might take some time to open the video and the thresholding menu. The default thresholds values should be adjusted in the ImageJ macro, until appropriate values are found. These values should then be used in the R functions / script, e.g., when calling the Locate_and_measure_particles function.

## Usage

```
check_threshold_values(
  to.data = par_to.data(),
  raw.video.folder = par_raw.video.folder(),
  ijmacs.folder = par_ijmacs.folder(),
  vid_select = 0,
  difference.lag = par_difference.lag(),
  thresholds = par_thresholds(),
  IJ.path = par_IJ.path(),
  memory = par_memory(),
  java.path = par_java.path()
)
```

## Arguments

| | |
|---|---|
| `to.data` | path to the working directory |

| | |
|---|---|
| raw.video.folder | directory with the raw video files |
| ijmacs.folder | directory where the check_trehsolds macro is saved |
| vid_select | video selected to find appropriate thresholds; default is the first video |
| difference.lag | numeric value specifying the offset between two frames of a video |
| thresholds | Numeric vector containing the min and max threshold values |
| IJ.path | path to ImageJ folder, containing the ij.jar executable |
| memory | numeric value specifying the amount of memory available to ImageJ (defaults to 512) |

---

`check_video_file_names` *Function to check that video files have extension cxd or avi, and that they are otherwise compatible.*

---

## Description

Checks the files in the raw data for the supported avi and cxd file formats and that file names do not contain periods except before the file type extension

## Usage

```
check_video_file_names(
  to.data = par_to.data(),
  raw.video.folder = par_raw.video.folder(),
  video.description.folder = par_video.description.folder(),
  video.description.file = par_video.description.file()
)
```

## Arguments

| | |
|---|---|
| to.data | path to the working directory |
| raw.video.folder | directory with the raw video files |

## Value

returns an error message and a list with unsupported files or names

`convert_cxd_to_avi` *Function to convert the video files to .avi format using lossless conversion*

## Description

Function uses bftools to convert `cxd` files to `avi` and ffmpeg to compress these to lossles avi

## Usage

```
convert_cxd_to_avi(
  cxd_file,
  avi_dir,
  compression_level = 6,
  ffmpeg = par_ffmpeg(),
  bfconvert = par_bfconvert(),
  showinf = par_showinf(),
  delete_cxd = FALSE,
  mc.cores = par_mc.cores()
)
```

## Arguments

| | |
|---|---|
| `cxd_file` | one or more `cxd` file to be converted or a directory with `.cxd` files. |
| `avi_dir` | directory for the converted `cxd` files and the metadata files |
| `compression_level` | compression level - defaults to 6. Smaller numbers: faster and larger, larger numbers (maximum 9) smaller and slower. |
| `ffmpeg` | execuable ffmpeg. May have to be including path. |
| `bfconvert` | executable bfconvert from bftools. May have to be including path. |
| `showinf` | executable showinf from bftools. May have to be including path. |
| `delete_cxd` | if `TRUE`, the `.cxd` file will be deleted after successful conversion. Default: `FALSE` |
| `mc.cores` | Number of cores to be used when more than one xd file is given. |

## Value

returns nothing (NULL)

`Create_folder_structure` *Create folder structure based on parameter*

## Description

Create folder structure based on parameter

## Usage

```
Create_folder_structure(
  to.data = par_to.data(),
  video.description.folder = par_video.description.folder(),
  raw.video.folder = par_raw.video.folder(),
  particle.data.folder = par_particle.data.folder(),
  trajectory.data.folder = par_trajectory.data.folder(),
  temp.overlay.folder = par_temp.overlay.folder(),
  overlay.folder = par_overlay.folder(),
  merged.data.folder = par_merged.data.folder(),
  ijmacs.folder = par_ijmacs.folder()
)
```

## Arguments

| | |
|---|---|
| `to.data` | path to the working directory |
| `video.description.folder` | directory containing the video description file |
| `raw.video.folder` | directory with the raw video files |
| `particle.data.folder` | directory to which the data is saved as a text file |
| `trajectory.data.folder` | directory containing the global trajectory data |
| `temp.overlay.folder` | temporary directory to save the overlay created with R |
| `overlay.folder` | directory where the overlay videos are saved |
| `merged.data.folder` | directory where the global database is saved |
| `ijmacs.folder` | directory for the macro to for ImageJ |

## Value

invisibly `TRUE`

`create_overlays` *Function to create a new video with the extracted trajectories overlayed onto the original video*

## Description

A function to overlay the extracted trajectories onto the original video, using plots created in R and then processed using ffmpeg; two visualization types are available

## Usage

```
create_overlays(
  to.data = par_to.data(),
  merged.data.folder = par_merged.data.folder(),
  raw.video.folder = par_raw.video.folder(),
  temp.overlay.folder = par_temp.overlay.folder(),
  overlay.folder = par_overlay.folder(),
  width = par_width(),
  height = par_height(),
  difference.lag = par_difference.lag(),
  type = "traj",
  predict_spec = FALSE,
  ffmpeg = "ffmpeg"
)
```

## Arguments

| | |
|---|---|
| `to.data` | path to the working directory |
| `merged.data.folder` | directory where the global database is saved |
| `raw.video.folder` | directory with the raw video files |
| `temp.overlay.folder` | temporary directory to save the overlay created with R |
| `overlay.folder` | directory where the overlay videos are saved |
| `width` | width of the raw video |
| `height` | height of the raw video |
| `difference.lag` | numeric value specifying the offset between two video frames to compute the difference image |
| `type` | string indicating the visualization type (i.e. 'label' or 'traj'): either the overlay is showing the trajectory ID and outlines the detected particle (type='label') or the whole trajectory remains plotted (type='traj'). |
| `predict_spec` | logical or character If `TRUE`, the Master.rds file must have a column called predict_spec, indicating the species to which the trajectory belongs. If a `character`, the Master.rds file must have a column called the value of the `character` value, indicating the species to which the trajectory belongs |

| | |
|---|---|
| ffmpeg | command to run ffmpeg. The default is ffmpeg. It can include a path. |

---

`create_overlays_subtitle` *Function to make overlays using the subtitle files,*

---

## Description

Function creates subtitle commands for every particle in every frame of the videos, using the x/y coordinates. Then ffmpeg is called to burn the overlay subtitles on, and save a compressed video as an mp4 file.

## Usage

```
create_overlays_subtitle(
  to.data = par_to.data(),
  merged.data.folder = par_merged.data.folder(),
  raw.video.folder = par_raw.video.folder(),
  temp.overlay.folder = par_temp.overlay.folder(),
  overlay.folder = par_overlay.folder(),
  label = "trajectory",
  ffmpeg = par_ffmpeg(),
  master = par_master(),
  overlay.type = "both",
  font_size = 24,
  circle_size = 120,
  crf = 23,
  gamma = 2,
  mc.cores = par_mc.cores()
)
```

## Arguments

| | |
|---|---|
| to.data | path to the working directory |
| merged.data.folder | directory where the global database is saved relative to the to.data directy |
| temp.overlay.folder | temporary directory to save the overlay subtitles (.ssa files) relative to the to.data directy |
| overlay.folder | directory where the overlay videos are saved relative to the to.data directly |
| label | column to be used to label the particle. Default is "trajectory", other useful might be "species" |
| ffmpeg | command to run ffmpeg. The default is par_ffmpeg(). It can include a path. |
| master | name of the master file. Defaults to par_master() |
| overlay.type | option for the overlays. Overlays can either be shown as "label", "circle" or "both" |

| | |
|---|---|
| font_size | size of the font for the labels. Default: 24 |
| circle_size | size of the circle. Default: 120 |
| crf | integer value between 1 to 51, where 1 means lossless, 17 is nearly visually lossless, 51 is worst quality. Default value is 23 |
| gamma | gamma correction. Value between 0.1 and 10. Default 2. see https://ffmpeg.org/ffmpeg-filters.html#eq for further info |
| mc.cores | number of cores toi be used for parallel execution. Defaults to par_mc.cores() |
| raw.avi.folder | path to the folder containing the converted and compressed .avi files relative to the to.data directy |

## Value

returns invisibly NULL

---

# create_overlays_subtitle_directory

*Function to create overlays for all aci files in a directory by using ffmpeg and subtitles*

---

## Description

Function creates subtitle commands for each particle in every frame of the videos, using the x/y coordinates. Then ffmpeg is called to burn the overlay subtitles in the video, and save a compressed video as an mp4 file.

## Usage

```
create_overlays_subtitle_directory(
  traj_data,
  avi_file_dir,
  crop = par_crop_pixels(),
  temp_overlay_folder = tempfile(),
  overlay_folder = ".",
  overlay_type = "both",
  label = "trajectory",
  ffmpeg = "ffmpeg",
  font_size = 24,
  circle_size = 120,
  crf = 23,
  gamma = 2,
  mc_cores = par_mc.cores()
)
```

### Arguments

| | |
|---|---|
| traj_data | object containing the trajectory data (usually from the Master file) |
| avi_file_dir | directory containing the input .avi files |
| temp_overlay_folder | temporary directory to save the overlay subtitles (.ssa files). Defaults to a temporaty directory. |
| overlay_folder | directory where the overlay video will be saved |
| overlay_type | option for the overlays. Overlays can either be shown as "label", "circle" or "both" |
| label | column to be used to label the particle. Default is "trajectory", other useful might be "species" |
| ffmpeg | command to run ffmpeg. The default is par_ffmpeg(). It can include a path. |
| font_size | size of the font for the labels. Default: 24 |
| circle_size | size of the circle. Default: 120 |
| crf | integer value between 1 to 51, where 1 means lossless, 17 is nearly visually lossless, 51 is worst quality. Default value is 23 |
| gamma | gamma correction. Value between 0.1 and 10. Default 2. see https://ffmpeg.org/ffmpeg-filters.html#eq for further info |
| mc_cores | number of cores toi be used for parallel execution. Defaults to par_mc.cores() |
| parameter | for croppoing rectangle to be drawn. An object as returned by par_crop_pixels() |

### Value

returns invisibly the stderr and stdout of the ffmpeg invocations

---

## create_overlays_subtitle_new

*Function to make overlays using the subtitle files,*

---

### Description

Function creates subtitle commands for every particle in every frame of the videos, using the x/y coordinates. Then ffmpeg is called to burn the overlay subtitles on, and save a compressed video as an mp4 file.

### Usage

```
create_overlays_subtitle_new(
  to.data = par_to.data(),
  merged.data.folder = par_merged.data.folder(),
  raw.video.folder = par_raw.video.folder(),
  temp.overlay.folder = par_temp.overlay.folder(),
  overlay.folder = par_overlay.folder(),
```

```
  label = "trajectory",
  ffmpeg = par_ffmpeg(),
  master = par_master(),
  overlay.type = "both",
  font_size = 24,
  circle_size = 120,
  crf = 23,
  gamma = 2,
  mc.cores = par_mc.cores()
)
```

## Arguments

| | |
|---|---|
| `to.data` | path to the working directory |
| `merged.data.folder` | directory where the global database is saved relative to the `to.data` directy |
| `temp.overlay.folder` | temporary directory to save the overlay subtitles (.ssa files) relative to the `to.data` directy |
| `overlay.folder` | directory where the overlay videos are saved relative to the `to.data` directly |
| `label` | column to be used to label the particle. Default is `"trajectory"`, other useful might be `"species"` |
| `ffmpeg` | command to run ffmpeg. The default is `par_ffmpeg()`. It can include a path. |
| `master` | name of the master file. Defaults to `par_master()` |
| `overlay.type` | option for the overlays. Overlays can either be shown as `"label"`, `"circle"` or `"both"` |
| `font_size` | size of the font for the labels. Default: 24 |
| `circle_size` | size of the circle. Default: 120 |
| `crf` | integer value between 1 to 51, where 1 means lossless, 17 is nearly visually lossless, 51 is worst quality. Default value is 23 |
| `gamma` | gamma correction. Value between 0.1 and 10. Default 2. see https://ffmpeg.org/ffmpeg-filters.html#eq for further info |
| `mc.cores` | number of cores toi be used for parallel execution. Defaults to `par_mc.cores()` |
| `raw.avi.folder` | path to the folder containing the converted and compressed .avi files relative to the `to.data` directy |

## Value

returns invisibly the stderr and stdout of the ffmpeg invocations

`create_overlays_subtitle_single` *Function to create a single overlay by using ffmpeg and subtitles*

## Description

Function creates subtitle commands for each particle in every frame of the videos, using the x/y coordinates. Then ffmpeg is called to burn the overlay subtitles in the video, and save a compressed video as an mp4 file.

## Usage

```
create_overlays_subtitle_single(
  traj_data,
  avi_file,
  crop = par_crop_pixels(),
  temp_overlay_folder = tempfile(),
  overlay_folder = ".",
  overlay_type = "both",
  label = "trajectory",
  ffmpeg = "ffmpeg",
  font_size = 24,
  circle_size = 120,
  crf = 23,
  gamma = 2,
  to_do = c("subtitles", "burnin")
)
```

## Arguments

| | |
|---|---|
| `traj_data` | object containing the trajectory data (usually from the Master file) |
| `avi_file` | input .avi file |
| `temp_overlay_folder` | temporary directory to save the overlay subtitles (.ssa files). Defaults to a temporaty directory. |
| `overlay_folder` | directory where the overlay video will be saved |
| `overlay_type` | option for the overlays. Overlays can either be shown as `"label"`, `"circle"` or `"both"` |
| `label` | column to be used to label the particle. Default is `"trajectory"`, other useful might be `"species"` |
| `ffmpeg` | command to run ffmpeg. The default is `par_ffmpeg()`. It can include a path. |
| `font_size` | size of the font for the labels. Default: 24 |
| `circle_size` | size of the circle. Default: 120 |
| `crf` | integer value between 1 to 51, where 1 means lossless, 17 is nearly visually lossless, 51 is worst quality. Default value is 23 |

| | |
|---|---|
| gamma | gamma correction. Value between 0.1 and 10. Default 2. see https://ffmpeg.org/ffmpeg-filters.html#eq for further info |
| to_do | what should be done. If "subtitles", only the subtitle files are created. If "burnin", the subtitles generated by "subtitles" are burned into the video file. Default is to do both. |
| parameter | for croppoing rectangle to be drawn. An object as returned by par_crop_pixels() |

## Value

returns invisibly the stderr and stdout of the ffmpeg invocation

---

`filter_data` *Function to filter trajectories based on movement and detection rate characteristics*

---

## Description

The function creates a dataframe containing all trajectories that are valid for further analysis by selecting on minimum net displacement, detection rate, trajectory length and the median step_length

## Usage

```
filter_data(
  raw_data = par_raw_data(),
  net_filter = par_net_filter(),
  duration_filter = par_duration_filter(),
  detect_filter = par_detect_filter(),
  median_step_filter = par_median_step_filter(),
  fps = par_fps()
)
```

## Arguments

| | |
|---|---|
| raw_data | dataframe containing the rawdata: X- and Y-coordinates, frame, file and trajectory name, morphology and movement metrics |
| net_filter | minimum net displacement to be considered a valid trajectory (in the length scale specified) |
| duration_filter | minimum duration to be considered a valid trajectory (in seconds) |
| detect_filter | minimum detection rate to be considered a valid trajectory (a proportion between 0 and 1) |
| median_step_filter | threshold such that half of the step lengths are above the specified value |
| fps | Frames Per Second of the video |

## Value

returns a dataset with all fixes of valid trajectories

## fix_crop_pixels *Fix croppimg parameter*

### Description

Fix croppimg parameter

### Usage

```
fix_crop_pixels(crop_pixels = par_crop_pixels())
```

### Arguments

```
crop_pixels
```

## get_delays_cxd *Extract delays from the* .cxd *file*

### Description

This function reads the metadata from the .cxd file and returns the delays between the images were taken.

### Usage

```
get_delays_cxd(file, showinf = par_showinf(), mc.cores = par_mc.cores())
```

### Arguments

| | |
|---|---|
| file | file name, of the .cxd file |
| mc.cores | Number of cores to be used when more than one xd file is given. |

### Value

a vector of the length of number of frames -1 specifying the delay between the images taken.

## get_duration_avi *Extract duration in seconds using ffmpeg*

### Description

The duration is based on using the output of `ffmpeg` and should therefore work for all ffmpeg supported video formats, but is only implemented for `.avi` files.

### Usage

```
get_duration_avi(file, ffmpeg = par_ffmpeg(), mc.cores = par_mc.cores())
```

### Arguments

| | |
|---|---|
| `file` | file name, of the video file |
| `mc.cores` | Number of cores to be used when more than one xd file is given. |

### Value

named vector with the duration (in seconds) of the video(s) and the names are the file names

## get_fps_avi *Extract fps (frames per second) using ffmpeg*

### Description

The fps is based on using the output of `ffmpeg` and should therefore work for all ffmpeg supported video formats, but at the moment only supported for `avi` files.

### Usage

```
get_fps_avi(file, ffmpeg = par_ffmpeg(), mc.cores = par_mc.cores())
```

### Arguments

| | |
|---|---|
| `file` | file name(s), of the video (`avi`) file(s) or directory in which the video files are |
| `mc.cores` | Number of cores to be used when more than one xd file is given. |

### Value

named vector with the fps of the video(s) and the names are the file names

## get_fps_cxd *Extract fps (frames per second) from the* .cxd *file*

### Description

The fps is based on the results of `get_delays_cxd(file)` by averaging these and calculating from this mean the fps.

### Usage

```
get_fps_cxd(file, showinf = par_showinf(), mc.cores = par_mc.cores())
```

### Arguments

| | |
|---|---|
| `file` | file name, of the `.cxd` file |

### Value

the fps (frames per second) of the video

## get_height_avi *Extract height in pixels using ffmpeg*

### Description

The height is based on using the output of `ffmpeg` and should therefore work for all ffmpeg supported video formats, but is only implemented for `.avi` files.

### Usage

```
get_height_avi(file, ffmpeg = par_ffmpeg(), mc.cores = par_mc.cores())
```

### Arguments

| | |
|---|---|
| `file` | file name, of the video file |
| `mc.cores` | Number of cores to be used when more than one xd file is given. |

### Value

named vector with the height (in pixels) of the video(s) and the names are the file names

## get_width_avi *Extract width in pixels using ffmpeg*

### Description

The width is based on using the output of `ffmpeg` and should therefore work for all ffmpeg supported video formats, but is only implemented for `.avi` files.

### Usage

```
get_width_avi(file, ffmpeg = par_ffmpeg(), mc.cores = par_mc.cores())
```

### Arguments

| | |
|---|---|
| `file` | file name, of the video file |
| `mc.cores` | Number of cores to be used when more than one xd file is given. |

### Value

named vector with the width (in pixels) of the video(s) and the names are the file names

## ij.bin *Return ImegeJ executable based on platform. Darwin, Windows and Linux are supported*

### Description

Return ImegeJ executable based on platform. Darwin, Windows and Linux are supported

### Usage

```
ij.bin()
```

### Value

full path of ewxecutable of ImageJ

### Examples

```
ij.bin()
```

`link_particles` *Function to link the particle coordinates through time*

## Description

The function takes the XY-coordinates provided by the ImageJ ParticleAnalyzer and uses a standalone version of the ImageJ MOSAIC plugin ParticleLinker to create trajectories. This requires some creation of temporary files, which are subsequently deleted.

## Usage

```
link_particles(
  to.data = par_to.data(),
  particle.data.folder = par_particle.data.folder(),
  trajectory.data.folder = par_trajectory.data.folder(),
  linkrange = par_linkrange(),
  disp = par_disp(),
  start_vid = 1,
  memory = par_memory(),
  to.particlelinker = par_to.particlelinker(),
  pixel_to_scale = par_pixel_to_scale(),
  fps = par_fps(),
  java.path = par_java.path()
)
```

## Arguments

| | |
|---|---|
| `to.data` | path to the working directory |
| `particle.data.folder` | directory where the ParticleAnalyzer output is saved (as text files) (temporary) |
| `trajectory.data.folder` | directory where the ParticleLinker is saved (as text files) (temporary???) |
| `linkrange` | numeric value passed to the ParticleLinker specifying the range of adjacent frames which are taken into account when a trajectory is re-constructed |
| `disp` | numeric value that specifies the maximum displacement of a given particle between two frames |
| `start_vid` | numeric value to indicate whether the linking should be started with a video other than the first |
| `memory` | numeric value specifying the max amount of memory allocated to the ParticleLinker (defaults to 512) |
| `to.particlelinker` | path to ParticleLinker jar file |
| `pixel_to_scale` | TODO |
| `fps` | Frames Per Second of the video |

### Value

Returns a single text file per video containing the X- and Y-coordinates, the frame and a trajectory ID. The files are than automatically merged into a data.table with the movement metrics for each fix appended to the original data (NB: movement metrics often need two (e.g. step length), sometimes even three (e.g., turning angles) fixes; fixes for which metrics cannot be calculated are padded with NA). The movement parameters are the step length, the step duration, the step speed (step length/step duration), the gross displacement as the cumulative sum of the step lengths, the net displacement between the first fix of a given trajectory and the current fix and finally the relative angle (turning angle) and absolute angle (in radians). For details on these metrics, please refer to a dedicated textbook (e.g. Turch (1998): Quantitative Analysis of Movement: Measuring and Modeling Population Redistribution in Animals and Plants, Sinauer Associates, Sunderland).

---

## load_parameter *Load parameter from* file

---

### Description

Load parameter from file

### Usage

```
load_parameter(file = "parameter.yaml")
```

### Arguments

| | |
|---|---|
| file | name of parameter file |

### Value

invisibly TRUE

---

## locate_and_measure_particles *Function to extract morphological measurements and X- and Y-coordinates for moving particles*

---

### Description

Function calls ImageJ software and its ParticleAnalyzer function to extract for each frame of the video several morphological descriptors and the X- and Y-coordinates of all moving particles. All videos in the raw.video.folder are analyses, separately.

## Usage

```
locate_and_measure_particles(
  to.data = par_to.data(),
  raw.video.folder = par_raw.video.folder(),
  particle.data.folder = par_particle.data.folder(),
  difference.lag = par_difference.lag(),
  min_size = par_min_size(),
  max_size = par_max_size(),
  thresholds = par_thresholds(),
  crop_pixels = par_crop_pixels(),
  IJ.path = par_IJ.path(),
  memory = par_memory(),
  ijmacs.folder = par_ijmacs.folder(),
  pixel_to_scale = par_pixel_to_scale(),
  java.path = par_java.path()
)
```

## Arguments

| | |
|---|---|
| `to.data` | path to the working directory |
| `raw.video.folder` | directory with the raw video files |
| `particle.data.folder` | directory to which the data is saved as a text file |
| `difference.lag` | numeric value specifying the offset between two video frames to compute the difference image. If 0, then no differencing applied. |
| `min_size` | minimum size for detection of particles |
| `max_size` | maximum size for detection of particles |
| `thresholds` | vector containing the min and max threshold values (defaults to c(10,255)) |
| `crop_pixels` | pixels to which the particle data should be cropped |
| `IJ.path` | path to ImageJ folder, containing the 'ij.jar' executable |
| `memory` | numeric value specifying the amount of memory available to ImageJ (defaults to 512) |
| `ijmacs.folder` | directory for the macro to for ImageJ |
| `pixel_to_scale` | TODO |

## Value

saves the output of the ParticleAnalyzer function of ImageJ as a text file in the output directory and then assembles the data into a single database called 'particle.rds'. This data.frame contains information about the following properties: the area (transversal cut), the mean, minimum and maximum of the grey value, the perimeter, width, length and angle with the dominant-axis of a fitted ellipse, and finally shape parameters such as circularity, aspect ratio, roundness and solidity. For details of the morphological output, please refer to http://rsbweb.nih.gov/ij/docs/guide/146-30.html

`merge_data` *Function to create global database containg morphology, trajectory, and video (e.g., experimental) information*

## Description

Merges the morphology data, the trajectory data with the video descriptions, which can / should contain the information on sampling units, video date and time, treatments and replicate etc. The files are merged by the use of the video file names. For the exact meaning of each of the columns, please refer to the locate_and_measure_particles() and link_particles() functions.

## Usage

```
merge_data(
  to.data = par_to.data(),
  particle.data.folder = par_particle.data.folder(),
  trajectory.data.folder = par_trajectory.data.folder(),
  video.description.folder = par_video.description.folder(),
  video.description.file = par_video.description.file(),
  merged.data.folder = par_merged.data.folder()
)
```

## Arguments

| | |
|---|---|
| `to.data` | path to the working directory |
| `particle.data.folder` | directory containing the global morphology data |
| `trajectory.data.folder` | directory containing the global trajectory data |
| `video.description.folder` | directory containing the video description file |
| `video.description.file` | name of the video description file |
| `merged.data.folder` | directory where the global database is saved |

## Value

saves the global database Master.rds to the merged.data.folder

## organise_link_data *Function to merge the output of the ParticleLinker into one large database*

### Description

Merge the trajectory data from the ParticleLinker into one data file

### Usage

```
organise_link_data(
  to.data = par_to.data(),
  trajectory.data.folder = par_trajectory.data.folder()
)
```

### Arguments

| | |
|---|---|
| to.data | path to the working directory |
| trajectory.data.folder | directory where the output of the ParticleLinker is saved |

### Value

saves the data containing the X- and Y coordinates of a given trajectory, the frame, the trajectory ID and the file name of the video from which the data was extracted to disk

## organise_particle_data *Function to merge the morphology and data on X- and Y-coordinates into one file for further processing*

### Description

This function merges the files containing morphology and coordinates (one for each video) into large dataset, and saves it to the directory where the single files are located

### Usage

```
organise_particle_data(
  to.data = par_to.data(),
  particle.data.folder = par_particle.data.folder(),
  pixel_to_scale = par_pixel_to_scale()
)
```

### Arguments

| | |
|---|---|
| to.data | path to the working directory |
| particle.data.folder | directory to which the data is saved as a text file |
| pixel_to_scale | TODO |

---

## par_bfconvert *parameter executable bfconvert of the bftools commandline tools*

---

### Description

parameter executable bfconvert of the bftools commandline tools

### Usage

```
par_bfconvert(value)
```

---

## par_classifier_constant *Name of classifier file for constant temperature treatment*

---

### Description

Name of classifier file for constant temperature treatment

### Usage

```
par_classifier_constant(value)
```

---

## par_classifier_increasing *Name of classifier file for increasing temperature treatment*

---

### Description

Name of classifier file for increasing temperature treatment

Name of classifiers,, i.e. list of classifiers to be used

### Usage

```
par_classifier_increasing(value)

par_classifiers(value)
```

## par_crop_pixels *Cropping of identified and measured particles.*

### Description

Cropping of identified and measured particles.

### Usage

```
par_crop_pixels(value)
```

### Details

This function returns a list with four elements, namely `xmin`, `xmax`, `ymin` and `ymax`. To set this parameter **all need to be set in a named list!** If a value is `NULL` (`~` in the parameter file), it is assumed to be the maximum or minimum value.

## par_detect_filter *parameter detect_filter*

### Description

parameter detect_filter

### Usage

```
par_detect_filter(value)
```

## par_difference.lag *parameter difference.lag*

### Description

parameter difference.lag

### Usage

```
par_difference.lag(value)
```

## par_disp *parameter disp*

### Description

parameter disp

parameter disp

### Usage

```
par_disp(value)

par_extrapolation.factor(value)
```

## par_duration_filter *parameter duration_filter*

### Description

parameter duration_filter

### Usage

```
par_duration_filter(value)
```

## par_ffmpeg *parameter executable ffmpeg*

### Description

parameter executable ffmpeg

### Usage

```
par_ffmpeg(value)
```

## par_fps *parameter fps*

### Description

parameter fps

### Usage

```
par_fps(value)
```

## par_height *parameter height*

### Description

parameter height

### Usage

```
par_height(value)
```

## par_IJ.path *parameter IJ.path*

### Description

parameter IJ.path

### Usage

```
par_IJ.path(value)
```

## par_ijmacs.folder *parameter ijmacs.folder*

### Description

parameter ijmacs.folder

### Usage

```
par_ijmacs.folder(value)
```

## par_java.path *parameter java.path*

### Description

parameter java.path

### Usage

```
par_java.path(value)
```

## par_linkrange *parameter linkrange*

### Description

parameter linkrange

### Usage

```
par_linkrange(value)
```

## par_master *Name of Master file*

### Description

Name of Master file

### Usage

```
par_master(value)
```

## par_max_area *parameter max_area*

### Description

parameter max_area

### Usage

```
par_max_area(value)
```

## par_max_size *parameter max_size*

### Description

parameter max_size

### Usage

```
par_max_size(value)
```

## par_mc.cores *Number of cores to be used for parallel processing. Defaults to 1, i.e. no parallel processing*

### Description

Number of cores to be used for parallel processing. Defaults to 1, i.e. no parallel processing

### Usage

```
par_mc.cores(value)
```

## par_mean_density *Name of Mean density per ml file*

### Description

Name of Mean density per ml file

### Usage

```
par_mean_density(value)
```

## par_median_step_filter *parameter median_step_filter*

### Description

parameter median_step_filter

### Usage

```
par_median_step_filter(value)
```

## par_memory *parameter memory*

### Description

parameter memory

### Usage

```
par_memory(value)
```

## par_merged.data. unfiltered.folder *parameter merged.data.unfiltered.folder*

### Description

parameter merged.data.unfiltered.folder

### Usage

```
par_merged.data.unfiltered.folder(value)
```

## par_merged.data.folder *parameter merged.data.folder*

### Description

parameter merged.data.folder

### Usage

```
par_merged.data.folder(value)
```

## par_min_area *parameter min_area*

### Description

parameter min_area

### Usage

```
par_min_area(value)
```

## par_min_size *parameter min_size*

### Description

parameter min_size

### Usage

```
par_min_size(value)
```

## par_morph_mvt *Name of Morph_mvt file*

### Description

Name of Morph_mvt file

### Usage

```
par_morph_mvt(value)
```

## par_net_filter *parameter net_filter*

### Description

parameter net_filter

### Usage

```
par_net_filter(value)
```

## par_overlay.folder *parameter overlay.folder*

### Description

parameter overlay.folder

### Usage

```
par_overlay.folder(value)
```

## par_particle *Name of organised particles file*

### Description

Name of organised particles file

### Usage

```
par_particle(value)
```

## par_particle.data.folder *parameter particle.data.folder*

### Description

parameter particle.data.folder

### Usage

```
par_particle.data.folder(value)
```

## par_pixel_to_scale *parameter pixel_to_scale*

### Description

parameter pixel_to_scale

### Usage

```
par_pixel_to_scale(value)
```

## par_raw.video.folder *parameter raw.video.folder*

### Description

parameter raw.video.folder

### Usage

```
par_raw.video.folder(value)
```

## par_showinf *parameter executable showinf of the bftools commandline tools*

### Description

parameter executable showinf of the bftools commandline tools

### Usage

```
par_showinf(value)
```

## par_species_tracked *Species tracked*

### Description

Species tracked

### Usage

```
par_species_tracked(value)
```

## par_temp.overlay.folder *parameter temp.overlay.folder*

### Description

parameter temp.overlay.folder

### Usage

```
par_temp.overlay.folder(value)
```

## par_template *Template function to assign value to parameter in the package wide cache*

### Description

assign the function to a new cvariable and the name of the function woll be used for the parameter name. e.g:

- fps <- par_template

### Usage

```
par_template(value)
```

### Arguments

| | |
|---|---|
| value | if missing, the value of the parameter will be returned, NULL if the parameter does not exist; if specified, the parameter will be set to the value |

### Value

the (new) value of the argument

## par_thresholds *parameter thresholds*

### Description

parameter thresholds

### Usage

```
par_thresholds(value)
```

## par_timeout *parameter timeout. If 0, no timeout*

### Description

parameter timeout. If 0, no timeout

### Usage

```
par_timeout(value)
```

## par_to.data *parameter to.data / the base folder*

### Description

parameter to.data / the base folder

### Usage

```
par_to.data(value)
```

## par_to.particlelinker *parameter to.particlelinker*

### Description

parameter to.particlelinker

### Usage

```
par_to.particlelinker(value)
```

## par_trajectory *Name of organised trajectory file*

### Description

Name of organised trajectory file

### Usage

```
par_trajectory(value)
```

## par_trajectory.data.folder *parameter trajectory.data.folder*

### Description

parameter trajectory.data.folder

### Usage

```
par_trajectory.data.folder(value)
```

## par_video.description.file *parameter video.description.file*

### Description

parameter video.description.file

### Usage

```
par_video.description.file(value)
```

## par_video.description.folder *parameter video.description.folder*

### Description

parameter video.description.folder

### Usage

```
par_video.description.folder(value)
```

## par_width *parameter width*

### Description

parameter width

### Usage

```
par_width(value)
```

## print_parameter *Print the bemovi parameter*

### Description

Print the bemovi parameter

### Usage

```
print_parameter(print_as_yaml = TRUE, echo = TRUE)
```

### Arguments

| | |
|---|---|
| print_as_yaml | Print in yaml formated text; ~ stands for NULL |
| echo | if TRUE print the parameter, if FALSE just return them as list |

### Value

invisible returns list of parameter for further processing

## save_parameter *Save parameter into* .yaml *file*

### Description

Save parameter into .yaml file

### Usage

```
save_parameter(file = "parameter.yaml")
```

### Arguments

| | |
|---|---|
| file | name of parameter file |

### Value

invisibly TRUE

summarize_trajectories *Function to summarize the morphology, movement and their variability on the trajectory level*

## Description

Takes the data comprising the information for each frame and calculates summary statistics such as mean and sd (for all morphology metrics) and mean, sd and min/max for some of the movement metrics along the trajectory. Values are rounded to the second decimal.

## Usage

```
summarize_trajectories(
  data,
  calculate.median = TRUE,
  write = FALSE,
  to.data = par_to.data(),
  merged.data.folder = par_merged.data.folder(),
  fps = par_fps(),
  video.description.folder = par_video.description.folder(),
  video.description.file = par_video.description.file()
)
```

## Arguments

| | |
|---|---|
| `data` | dataframe with the information on morphology and movement for each frame |
| `calculate.median` | logical value to indicate whether the median/IQR or the mean/SD summaries should be calculated for the morphology |
| `write` | logical argument to indicate whether aggregated information should be saved to disk |
| `to.data` | path to the working directory |
| `merged.data.folder` | directory where the global database is saved |
| `fps` | Frames Per Second of the video |
| `video.description.folder` | folder in which video descriptions are found |
| `video.description.file` | name of the video description file |

## Value

returns a data.table with the aggregated morphology and movement information for each trajectory

# Index