DryMass Documentation

Release 0.10.2

Paul Müller

Mar 23, 2020

Contents

DryMass is a Python library and a command-line tool for manipulating and analyzing quantitative phase microscopy images of cell-sized objects. This is the documentation of DryMass version 0.10.2.

CHAPTER 1

Introduction

1.1 What is DryMass?

DryMass is a tool in quantitative phase imaging (QPI) for the extraction of physical properties such as volume, refractive index, and dry mass of pure phase objects. Features include

- automated extraction of meta data (e.g. wavelength, acquisition time) from experimental data files,
- automated detection of phase object positions (cells, beads, lipid droplets),
- elaborate methods for phase image background correction,
- determination of dry mass for biological cells, or
- extraction of refractive index and radius for spherical phase objects such as liquid droplets, microgel beads, or cells.

1.2 What is quantitative phase imaging?

Quantitative phase imaging (QPI) is a 2D imaging technique that quantifies the phase retardation of a wave traveling through a specimen. For instance, digital holographic microscopy (DHM) [KvB07] can be used to record the quantitative phase image of biological cells, yielding the optical density from which the *dry mass* or the refractive index (RI) can be computed. Another example is electron holography [LL02] which can be used to visualize p-n junctions due to the different electronic potentials in the doped semiconductors. DryMass was designed for the analysis of single cells (typical units for distance [µm] and wavelength [nm]), but the concepts used apply to both methods.

1.3 What are typical use cases of DryMass?

The focus of DryMass is the analysis of large data sets (e.g. > 10 cells). Here are a few examples:

- You have recorded several digital holograms and would like to extract phase and amplitude from them. When analyzing experimental data, DryMass automatically converts holograms (stored as .tif files) to phase and intensity image series, .tif files that can be opened with e.g. Fiji. You also have several options for automated phase/amplitude background correction.
- You are interested in characterizing microgel beads and recorded several quantitative phase images, each of them containing a few well-separated beads. By tuning only a few parameters, such as the expected specimen size, the phase image background correction method, or the scattering model, DryMass determines the position of the beads, and yields accurate values for refractive index, size, and dry mass for each bead.
- You would like to monitor cell dry mass over time. If the cells are well-separated, this task is trivial with DryMass. If in addition, the cells are spherical (e.g. suspended cells), then DryMass can also compute accurate values for mean refractive index and cell size.

1.4 How should I cite DryMass?

If you are using DryMass in a scientific publication, please cite it with:

```
(...) using DryMass version X.X.X (available at
https://pypi.python.org/pypi/drymass).
```

or in a bibliography

```
Paul Müller (2018), DryMass version X.X.X: Phase image analysis
[Software]. Available at https://pypi.python.org/pypi/drymass.
```

and replace `X.X.X` with the version of DryMass that you used.

Furthermore, several ideas that DryMass builds upon have been described and published in scientific journals:

- Retrieval of RI and radius using the OPD edge-detection approach is described in [SSM+15] and [SSM+16] (`method="edge"` and `model="projection"` in the `[sphere]` section of the *Configuration*).
- Retrieval of RI and radius by fitting 2D models (OPD projection, Rytov approximation, systematically corrected Rytov approximation, Mie) to phase images is described in [MSG+18]. (`method="image"` in the `[sphere]` section of the *Configuration*).

CHAPTER 2

Theory Notes

2.1 Computation of cell dry mass

The concept of cell dry mass computation was first introduced by Barer [Bar52]. The dry mass m of a biological cell is defined by its non-aqueous fraction $f(x, y, z)$ (concentration or density in g/L), i.e. the number of grams of protein and DNA within the cell volume (excluding salts).

$$m = \iiint f(x, y, z)\, dxdydz$$

The assumption of dry mass computation in QPI is that $f(x, y, z)$ is proportional to the RI of the cell $n(x, y, z)$ with a proportionality constant called the refraction increment α (units [mL/g])

$$n(x, y, z) = n_{\text{intra}} + \alpha f(x, y, z)$$

with the RI of the intracellular fluid n_{intra}, a dilute salt solution. These two equations can be combined to

$$m = \frac{1}{\alpha} \cdot \iiint (n(x, y, z) - n_{\text{intra}})\, dxdydz. \tag{2.1}$$

In QPI, the RI is measured indirectly as a projected quantitative phase retardation image $\phi(x, y)$.

$$\phi(x, y) = \frac{2\pi}{\lambda} \int (n(x, y, z) - n_{\text{med}})\, dz$$

with the vacuum wavelength λ of the imaging light and the refractive index of the cell-embedding medium n_{med}. Integrating the above equation over the detector area (x, y) yields

$$\iint \phi(x, y)\, dxdy = \frac{2\pi}{\lambda} \iiint (n(x, y, z) - n_{\text{med}})\, dxdydz \tag{2.2}$$

If the embedding medium has the same refractive index as the intracellular solute ($n_{\text{med}} = n_{\text{intra}}$), then equations (2.1) and (2.2) can be combined to

$$m_{\text{med=intra}} = \frac{\lambda}{2\pi\alpha} \cdot \iint \phi(x, y)\, dxdy.$$

For a discrete image, this formula simplifies to

$$m_{\text{med=intra}} = \frac{\lambda}{2\pi\alpha} \cdot \Delta A \cdot \sum_{i,j} \phi(x_i, y_j) \tag{2.3}$$

with the pixel area ΔA and a pixel-wise summation of the phase data.

2.2 Relative and absolute dry mass

If however the medium surrounding the cell has a different refractive index ($n_{\text{med}} \neq n_{\text{intra}}$), then the phase ϕ is measured relative to the RI of the medium n_{med} which causes an underestimation of the dry mass if $n_{\text{med}} > n_{\text{intra}}$. For instance, a cell could be immersed in a protein solution or embedded in a hydrogel with a refractive index of $n_{\text{med}} = n_{\text{intra}} + 0.002$. For a spherical cell with a radius of 10µm, the resulting dry mass is underestimated by 46pg. Therefore, it is called "relative dry mass" m_{rel}.

$$m_{\text{rel}} = \frac{\lambda}{2\pi\alpha} \cdot \iint \phi(x, y)\, dxdy,$$

If the imaged phase object is spherical with the radius R, then the "absolute dry mass" m_{abs} can be computed by splitting equation (2.1) into relative mass and suppressed spherical mass.

$$\begin{aligned} m_{\text{abs}} &= \frac{1}{\alpha} \cdot \iiint (n(x, y, z) - n_{\text{med}} + n_{\text{med}} - n_{\text{intra}})\, dxdydz \\ &= m_{\text{rel}} + \frac{4\pi}{3\alpha} R^3 (n_{\text{med}} - n_{\text{intra}}) \end{aligned}$$

For a visualization of the deviation of the relative dry mass from the actual dry mass for spherical objects, please have a look at the *relative vs. absolute dry mass example*.

2.3 Range of validity

Variations in the refraction increment may occur and thus the above considerations are not always valid. For a detailed discussion of the variables that affect the refraction increment, please see [BJ54].

2.3.1 Dependency on imaging wavelength

Barer and Joseph measured the refraction increment of several proteins in dependence of wavelength. In general, short wavelengths (366nm) yield values close to 0.200mL/g while long wavelengths (656nm) yield smaller values close to 0.180mL/g (table 3 in [BJ54]).

2.3.2 Dependency on protein concentration

The refraction increment has been reported to be linear for a wide range of protein concentrations. Barer and Joseph found that bovine serum albumin exhibits a linear refraction increment up to its limit of solubility (figure 2 in [BJ54]). They additionally received a personal communication stating that this is also the case for gelatin.

2.3.3 Dependency on pH, temperature, and salts

The refraction increment is little dependent on pH, temperature, and salts [BJ54].

2.3.4 Refraction increment and the mass of cells

Dry mass and actual mass of a cell differ by the weight of the intracellular fluid. This weight difference is defined by the volume of the cell minus the volume of the protein and DNA content. While it seems to be difficult to define a partial specific volume (PSV) for DNA, there appears to be a consensus regarding the PSV of proteins, yielding approximately 0.73mL/g (see e.g. reference [Bar57] as well as [HGC94] and question 843 of the O-manual referring to it). For example, the protein and DNA of a cell with a radius of 10µm and a dry mass of 350pg (cell volume 4.19pL, average refractive index 1.35) occupy approximately 0.73mL/g · 350pg = 0.256pL (assuming the PSV of protein and DNA are similar). Therefore, the actual volume of the intracellular fluid is 3.93pL (94% of the cell volume) which is equivalent to a mass of 3.93ng resulting in a total (actual) cell mass of 4.28ng. Thus, the dry mass of this cell makes up approximately 10% of its actual mass which leads to a total mass that is about 2% heavier than the equivalent volume of pure water (4.19ng).

2.4 Default parameters in DryMass

- **The default refraction increment** is $\alpha = 0.18$mL/g, as suggested for cells based on the refraction increment of cellular constituents by references [BJ54] and [Bar53]. The refraction increment can be manually set using the *configuration* key "refraction increment" in the "sphere" section.
- **The default refractive index of the intracellular fluid** in DryMass is assumed to be $n_{\text{intra}} = 1.335$, an educated guess based on the refractive index of phosphate buffered saline (PBS), whose osmolarity and ion concentrations match those of the human body.

CHAPTER 3

Getting started

3.1 Installing DryMass

DryMass is written in pure Python and supports Python version 3.6 and later. DryMass depends on several other scientific Python packages, including:

- numpy,
- scikit-image (segmentation).
- qpimage (phase data manipulation),
- qpformat (file formats),
- qpsphere (refractive index analysis, segmentation),

To install DryMass, use one of the following methods (package dependencies will be installed automatically):

- **from PyPI:** `pip install drymass`
- **from sources:** `pip install .` or `python setup.py install`

3.1.1 Upgrade

If you have installed an older version of DryMass and wish to upgrade to the latest version, use

`pip uninstall drymass` followed by

`pip install drymass`.

If you wish to install a specific version of DryMass (e.g. 0.3.0), use

`pip install 'drymass==0.3.0'`.

3.1.2 Known issues

- If you try to install from PyPI and get an error message similar to

  "Could not find a version that satisfies the requirement drymass (from versions:)
  No matching distribution found for drymass",

  please make sure that you are using Python version 3.6 or later with `python --version`. If that is already the case, please run `pip -vvv install drymass` and create an issue with the error messages (e.g. as a screenshot) that you get.
- If you are using Windows and the installation fails because *scikit-image* cannot be installed, e.g.

  " Rolling back uninstall of scikit-image
  Command python.exe [...] –compile failed with error code 1 in [...]/scikit-image",

  and you are using the Anaconda Python distribution, please install *scikit-image* via `conda install scikit-image`. If you are not using Anaconda, you can install one of the wheels provided by Christoph Gohlke (download e.g. "scikit_image-0.14.0-cp36-cp35m-win_amd64.whl" if you have installed the 64bit version of Python 3.6, navigate to the download directory and run `pip install scikit_image-0.14.0-cp35-cp36m-win_amd64.whl`).

3.2 Command-line interface

DryMass comes with a command line interface (CLI). To use the CLI, a command shell is required, such as the command prompt Cmd.exe on Windows, or Terminal.app on MacOS. Please note that if DryMass is installed in a virtual environment, then the DryMass CLI is only available if this environment is activated.

3.2.1 Basic commands

dm_convert

This command converts experimental data on disk to the TIF file format for use with Fiji/ImageJ and to the hdf5-based qpimage data file format. The experimental data files are loaded with the qpformat library, which supports several quantitative phase imaging file formats. If a specific format is not supported, please create an issue at the qpformat issue page. A typical use case of dm_convert on Windows is

```
dm_convert "d:\\data\path\to\experiment"
```

which is equivalent to

```
d:
cd "data\path\to"
dm_convert experiment
```

If this command is run initially for an experimental data set, the user is asked to enter or confirm imaging wavelength and detector pixel size. Then, a new directory `d:\\data\path\to\experiment_dm` is created with the following files:

drymass.cfg the user-editable *drymass configuration file* which is used in subsequent analysis steps

sensor_data.h5 the experimental data (including meta data) in the hdf5-based qpimage data file format

sensor_data.tif the experimental phase and amplitude series data as a tif file, importable in Fiji/ImageJ

Note that it is possible to edit the *drymass.cfg* file and to re-run the dm_convert command (or any other of the commands below) with these updated parameters.

dm_extract_roi

This command automatically finds and extracts regions of interest (ROIs) and performs an automated background correction for single-cell analysis. The usage is the same as that of dm_convert:

```
dm_extract_roi "d:\\data\path\to\experiment"
```

The command dm_extract_roi automatically runs dm_convert if it has not been run before. If ROI detection fails, the search parameters have to manually be updated in the *drymass configuration file*. The most important parameter is the diameter of the specimen in microns ("*size um*" in the *specimen* section); all other parameters are defined in the *roi* section. Note that the default parameters for the *roi* section are not written to the configuration file until dm_extract_roi is run. The following files are created by dm_extract_roi:

roi_data.h5 the extracted, background-corrected ROI data (including meta data) in the hdf5-based qpimage data file format

roi_data.tif the extracted, background-corrected ROI data as a tif file, importable in Fiji/ImageJ

roi_slices.txt the locations of the ROIs found as a txt file

sensor_roi_images.tif rendered sensor phase images with labeled ROIs; only created if "*roi images*" is set to "*True*" in the *output* section of the *drymass configuration file*

dm_analyze_sphere

This command is used for the analysis of spherical phase objects such as liquid droplets, beads, or suspended cells. The basic principle is thoroughly described in reference [SSM+16]. In short, this approach assumes that the objects found with dm_extract_roi are homogeneous and spherical which allows to extract parameters such as radius and refractive index from a single phase image (as opposed to tomographic approaches that require an acquisition of multiple phase images from different directions). The parameters for the sphere analysis, such as analysis method and scattering model, are defined in the *sphere* section of the *drymass configuration file*. For an overview of the available models, please refer to the qpsphere docs. The following files are created by dm_analyze_sphere (*METHOD* is the analysis method and *MODEL* is the scattering model defined in *drymass.cfg*):

sphere_METHOD_MODEL_data.h5 the quantitative sphere simulation data using *MODEL* with the parameters obtained with the combination of *METHOD* and *MODEL* for each of the ROIs obtained with dm_extract_roi in the hdf5-based qpimage data file format

sphere_METHOD_MODEL_images.tif rendered phase and intensity images of the input ROIs and the corresponding simulation, a difference, and a line plot through the phase image for visual inspection as a tif file, importable in Fiji/ImageJ

sphere_METHOD_MODEL_statistics.txt the analysis results, including refractive index, radius, and *relative and absolute dry mass* as a text file.

3.2.2 Advanced usage

Profile management with dm_profile

If some of the parameters (e.g. pixel size or wavelength) are not stored with the experimental data, DryMass will ask the user to enter these in the command prompt. This process can be time-consuming, especially if a recursive analysis is performed (see below). To simplify the analysis in such cases, DryMass has the command `dm_profile`, which allows to store existing DryMass configuration files in a local library and use them to analyze data.

```
# add a profile named "preset2018a"
dm_profile add preset2018a "d:\\data\path\to\experiment_dm\drymass.cfg"
# list all profiles within the local library (name and path will be shown)
dm_profile list
# remove the profile "preset2018a"
dm_profile remove preset2018a
# export all local profiles to a folder
dm_profile export "d:\\exported_profiles"
```

To use a profile stored in the local library for an analysis, simply pass its name with the command-line parameter `--profile`:

```
dm_extract_roi --profile preset2018a "d:\\data\path\to\another\experiment"
```

Alternatively, a configuration file may also be specified without adding it to the local library:

```
dm_extract_roi --profile "d:\\data\path\to\experiment_dm\drymass.cfg"
↪"d:\\data\path\to\another\experiment"
```

Note that the parameters in the profile are merged with the parameters in the configuration file of the analyzed data. In other words, if a parameter is missing in the profile, the parameter of the existing *drymass.cfg* is used. If it is not in *drymass.cfg* the default value is used.

Recursive analysis

By default, the basic analysis commands only accept a single measurement as an argument. If there are several measurements, e.g.

- `d:\\data\path\to\experiments\1`
- `d:\\data\path\to\experiments\2`
- `d:\\data\path\to\experiments\3`
- `d:\\data\path\to\experiments\4`
- `d:\\data\path\to\experiments\5`
- ...

then the command-line parameter `--recursive` can be used:

```
dm_extract_roi --recursive "d:\\data\path\to\experiments"
```

The `--recursive` parameter can also be combined with the `--profile` parameter, which allows for a largely automated analysis pipeline:

```
dm_extract_roi --recursive --profile preset2018a "d:\\data\path\to\experiments"
```

3.3 Configuration file

The DryMass configuration file *drymass.cfg* is located in the root of the output folder ("*_dm*" appended to the data path). The configuration file is divided into sections.

3.3.1 [bg] Background correction

DryMass uses the Python library qpimage for background correction. For detailed information on the algorithms (and the corresponding keyword arguments) used, please see `qpimage.bg_estimate`.

- **amplitude binary threshold** = None (*`float_or_str`*) – Binary image threshold value or method
  If not *None*, defines either a threshold for background segmentation or a method name in *`drymass.threshold.available_thresholds`*.
- **amplitude border perc** = 10 (`float`) – Amplitude bg border region to analyze [%]
  Set to 0 to disable.
- **amplitude border px** = 5 (`int`) – Amplitude bg border region to analyze [px]
  Set to 0 to disable.
- **amplitude data** = None (*`int_or_path`*) – Amplitude bg correction file or index
  Image indexing starts with 1. If a file, either specify the full path or the relative path to the directory containing the input data.
- **amplitude mask sphere** = None (`float`) – Circular mask for amplitude bg correction
  If not *None*, a mask is used for background correction. A value of '1.0' results in a mask with the radius as determined with the edge-detection approach from phase [sic]. Set this to '1.1' to exclude peripheral phase values. If the amplitude border values or binary thresholds are set, the intersection of the resulting masks is used for background correction. For procedural details see `qpsphere.cnvnc.bg_phase_mask_for_qpi()`.
- **amplitude offset** = mean (*`lcstr`*) – Amplitude bg correction offset method
  Valid values are defined in `qpimage.bg_estimate.VALID_FIT_OFFSETS`.
- **amplitude profile** = tilt (*`lcstr`*) – Amplitude bg correction profile method
  Valid values are defined in `qpimage.bg_estimate.VALID_FIT_PROFILES`.
- **enabled** = True (*`fbool`*) – Enable bg correction globally
  Set to *False* to disable background correction.
- **phase binary threshold** = True (*`fbool`*) – Enable bg correction globally
  Set to *False* to disable background correction.
- **phase border perc** = 10 (`float`) – Phase bg border region to analyze [%]
  Set to 0 to disable.
- **phase border px** = 5 (`int`) – Phase bg border region to analyze [px]
  Set to 0 to disable.
- **phase data** = None (*`int_or_path`*) – Phase bg correction file or index
  Image indexing starts with 1. If a file, either specify the full path or the relative path to the directory containing the input data.
- **phase mask sphere** = None (`float`) – Circular mask for phase bg correction
  If not *None*, a mask is used for background correction. A value of '1.0' results in a mask with the radius as determined with the edge-detection approach from phase. Set this to '1.1' to exclude peripheral phase values.

If the phase border values or binary thresholds are set, the intersection of the resulting masks is used for background correction. For procedural details see `qpsphere.cnvnc.bg_phase_mask_for_qpi()`.

- **phase offset** = mean (*lcstr*) – Phase bg correction offset method
  Valid values are defined in `qpimage.bg_estimate.VALID_FIT_OFFSETS`.
- **phase profile** = tilt (*lcstr*) – Phase bg correction profile method
  Valid values are defined in `qpimage.bg_estimate.VALID_FIT_PROFILES`.

3.3.2 [holo] Hologram analysis

These parameters tune the analysis of off-axis hologram data (if applicable). The parameters shown are passed to `qpimage.holo.get_field()`.

- **filter name** = disk (`str`) – Filter name for sideband isolation
- **filter size** = 0.3333333333333333 (`float`) – Filter size (fraction of the sideband frequency)
- **sideband** = 1 (*floattuple_or_one*) – Sideband ±1 or frequency coordinates

3.3.3 [meta] Image meta data

This section contains meta data of the experiment.

- **medium index** = None (`float`) – Refractive index of the surrounding medium
- **pixel size um** = None (`float`) – Detector pixel size [µm]
- **wavelength nm** = None (`float`) – Imaging wavelength [nm]

3.3.4 [output] Supplementary data output

This section defines what additional data are written to disk.

- **roi images** = True (*fbool*) – Rendered phase images with ROI location
- **sphere images** = True (*fbool*) – Phase/Intensity images for sphere analysis
- **sensor tif data** = True (*fbool*) – Phase/Amplitude sensor tif data

3.3.5 [roi] Extraction of regions of interest

The extraction of ROIs is done in *drymass.extractroi.extract_roi()*.

- **dist border px** = 10 (`int`) – Minimum distance of objects to image border [px]
- **eccentricity max** = 0.7 (`float`) – Allowed maximal eccentricity of the specimen
- **enabled** = True (*fbool*) – Perform automated search for ROIs
  If set to *False*, the file 'roi_slices.txt' must contain ROIs.
- **exclude overlap px** = 30.0 (`float`) – Allowed distance between two objects [px]
- **force** = None (*tupletupleint*) – Force ROI coordinates (x1,x2,y1,y2) [px]
- **ignore data** = None (*strlist_vsort*) – Exclude images and ROIs from the analysis
  To exclude individual ROIs found in a previous run, simply type the ROI index, i.e. 'exclude = 1.0, 2.2'. You may additionally exclude a full sensor image (e.g. image 3) with 'exclude = 1.0, 2.2, 3'.

- **pad border px** = 40 (`int`) – Padding of object regions [px]
- **size variation** = 0.5 (*`float01`*) – Allowed variation relative to specimen size
- **threshold** = li (*`float_or_str`*) – Threshold for phase data segmentation
  The threshold is defined via a name or as a number in [rad]. Valid names are given in *`drymass.threshold.available_thresholds`*.

3.3.6 [specimen] Specimen parameters

Prior information about the analyzed object(s).

- **size um** = 10 (`float`) – Approximate diameter of the specimen [µm]
  This is used as the initial value for the sphere analysis.

3.3.7 [sphere] Sphere-based image analysis

Retrieval of refractive index and radius is done with the Python module qpsphere. The parameters either apply to `qpsphere.edgefit.contour_canny()` or to `qpsphere.imagefit.alg.match_phase()`, depending on which analysis approach is used.

- **edge coarse** = 0.4 (`float`) – Coarse edge detection filter size
- **edge fine** = 0.1 (`float`) – Fine edge detection filter size
- **edge clip radius min** = 0.9 (`float`) – Interior edge point filtering radius
- **edge clip radius max** = 1.1 (`float`) – Exterior edge point filtering radius
- **edge iter** = 20 (`int`) – Maximum number iterations for coarse edge detection
- **image fit range position** = 0.05 (`float`) – Fit interpolation range for radius
- **image fit range radius** = 0.05 (`float`) – Fit interpolation range for radius
- **image fit range refractive index** = 0.1 (`float`) – Fit interpolation range for refractive index
- **image fix phase offset** = True (*`fbool`*) – Fix the simulation background phase to zero
- **image iter** = 100 (`int`) – Maximum number of iterations for image fitting
- **image stop delta position** = 1 (`float`) – Stopping criterion for position
- **image stop delta radius** = 0.001 (`float`) – Stopping criterion for radius
- **image stop delta refractive index** = 0.0005 (`float`) – Stopping criterion for refractive index
- **image verbosity** = 1 (`int`) – Verbosity level of image fitting algorithm
- **method** = edge (*`lcstr`*) – Method for determining sphere parameters
  Valid values are 'edge' (edge-detection approach) or 'image' (2D phase image fitting).
- **model** = projection (*`lcstr`*) – Physical sphere model
  Valid values are defined in `qpsphere.models.available`. If *method=edge*, then *model* must be set to *projection*. If *method=image*, setting *model* to *rytov-sc* has the best trade-off between accuracy and speed.
- **refraction increment** = 0.18 (`float`) – Refraction increment [mL/g]
- **radial inclusion factor** = 1.2 (`float`) – Radial inclusion factor for dry mass computation

3.4 Troubleshooting

3.4.1 Freezing / no output

If the *CLI* freezes and you see a blinking cursor for a very long time without any files being created or changed in the output directory (*_dm* appended to the input data set):

1. Remove all files in the output directory, **except** for *drymass.cfg*.
2. Run the analysis again.

3.4.2 None of the above

Please create an issue on GitHub with a screenshot (or copy-paste) of the error message.

CHAPTER 4

Tutorials

4.1 T1: Bead analysis (CLI)

4.1.1 Introduction

Microgel beads are transparent, homogeneous, and spherical objects that are ideal test objects for quantitative phase imaging. The DryMass command *dm_analyze_sphere* can estimate the average refractive index of such homogeneous objects. This is a short tutorial that will reproduce the data presented in supplementary figure 2a of reference [SCG+17].

4.1.2 Prerequisites

For this tutorial, you need:

- Python 3.6 or above and DryMass version 0.6.0 or above (see *Installing DryMass*)
- Fiji or Windows Photo Viewer (for data visualization)
- Experimental dataset: QLSR_PAA_beads.zip [MSGG19]

4.1.3 Execute dm_analyze_sphere

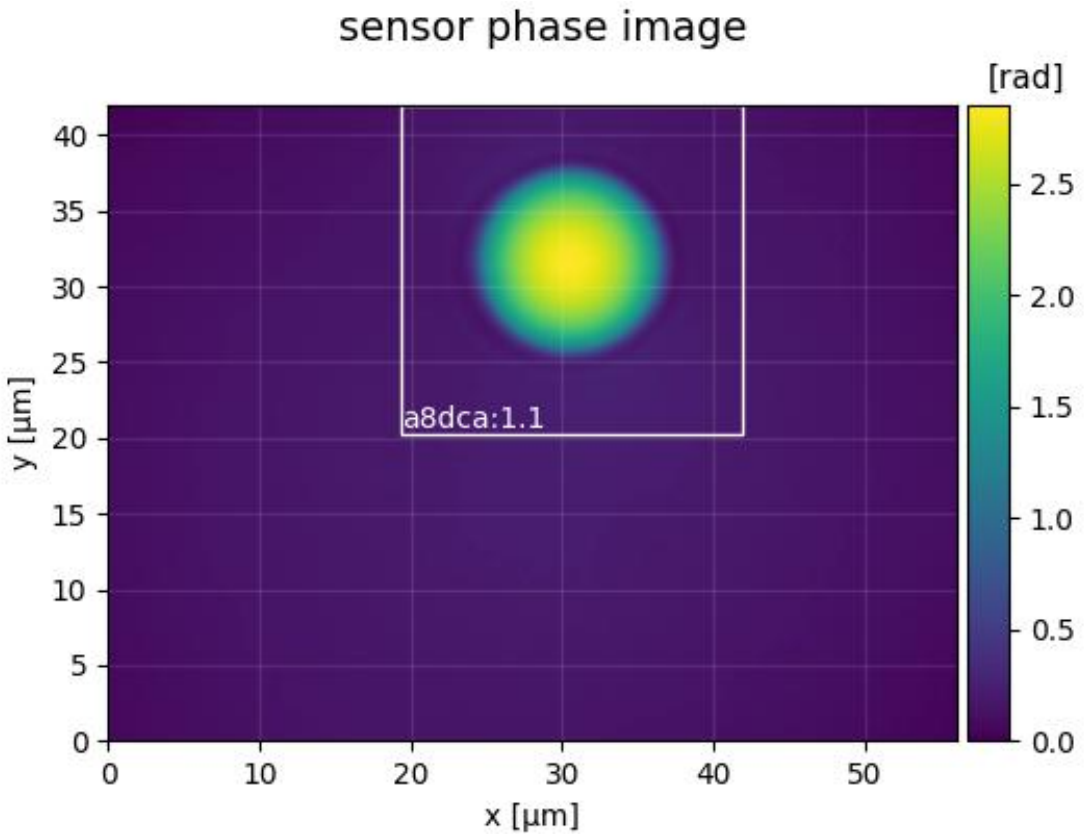
DryMass comes with a *Command-line interface* (CLI) which is made available after the installation. We will use the DryMass command *dm_analyze_sphere* to extract the refractive index values of a population of microgel beads. Using the command shell of your operating system, navigate to the location of QLSR_PAA_beads.zip and execute the command `dm_analyze_sphere` with `QLSR_PAA_beads.zip` as an argument. You will be prompted for the refractive index of the surrounding medium (1.335), the detector pixel size in microns (0.14), and the wavelength in nanometers (647). Simply type in these values (press the *Enter* key to let DryMass acknowledge each input). On Windows, this will look similar to this:

DryMass has created a directory called `QLSR_PAA_beads.zip_dm` (the input argument with *_dm* appended) which contains the following files

- **drymass.cfg**: DryMass *Configuration file*
- **roi_data.h5**: regions of interest (ROIs)
- **roi_data.tif**: phase and amplitude data of the ROIs as a tif file
- **roi_slices.txt**: positions of the ROIs
- **sensor_data.h5**: full sensor QPI data
- **sensor_data.tif**: full sensor phase and amplitude data as a tif file
- **sensor_roi_images.tif**: plotted full sensor phase images with ROIs
- **sphere_edge_projection_data.h5**: simulated (projection) phase and amplitude data
- **sphere_edge_projection_images.tif**: visualization of the sphere analysis of the ROIs as a tif file
- **sphere_edge_projection_statistics.txt**: sphere analysis results as a text file

4.1.4 Examine the results

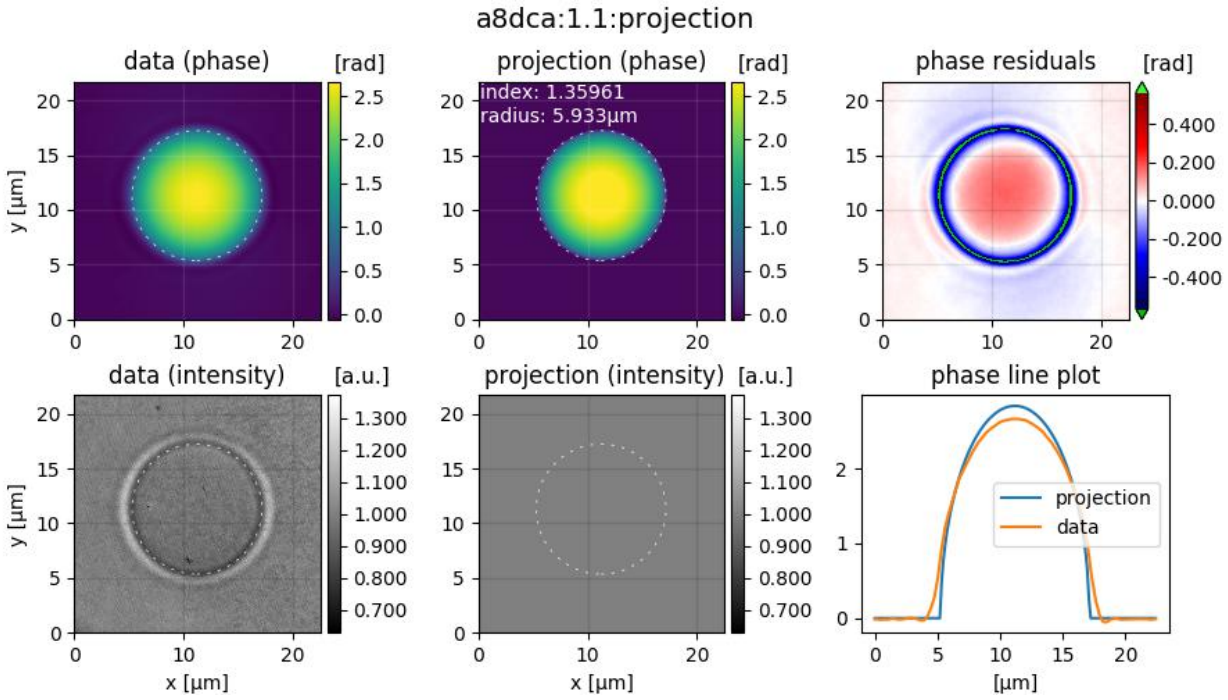
Let's have a look at *sensor_roi_images.tif* (using Fiji or Windows Photo Viewer). This is the first image stored in the tif file:



It shows the full sensor phase image of the first bead. The white rectangle indicates the ROI that was found by DryMass, labeled with the identifier *a8dca:1.1*. The file *sensor_roi_images.tif* allows you to check that DryMass has

correctly found the objects that you are interested in. If the beads were not detected correctly, we would probably have to adjust the size parameter in the *Configuration file* (see also *dm_extract_roi*).

It appears that DryMass has correctly found all beads with the default settings. Next, open the file *sphere_edge_projection_images.tif*. The identifier of the first ROI is shown at the top, the first column contains phase and intensity of the experimental data, the second column contains the modeled data (with refractive index and radius used for the simulation), and the third column shows the phase-difference as well as line plots through the phase images.



Note that the modeled intensity image is all-one, because the projection model only models the optical thickness and thus only affects the phase data. Also, note that the phase-difference image between data and model only has small deviations in the background phase. If the background phase was not flat, we would have to modify the *background correction*.

4.1.5 Post-processing

A closer examination of the phase-difference images shows that there seem to be either deformed beads or imaging artifacts in the images with the identifiers (prepend *a8dca:*): *4.1, 7.1, 24.1, 26.1, 27.1, 35.1, 36.1, 39.1, 40.1, 51.1, 52.1, 55.1, 58.1, 60.1, 64.1, 67.1,* and *71.1*. Due to their asymmetry we ignore these images in our analysis by editing the configuration file:

```
[roi]
ignore data = 4.1, 7.1, 24.1, 26.1, 27.1, 35.1, 36.1, 39.1, 40.1, 51.1, 52.1, 55.1,␣
↪58.1, 60.1, 64.1, 67.1, 71.1
```

After executing `dm_analyze_sphere` again, we can load the statistics file *sphere_edge_projection_statistics.txt* into a statistical analysis application and compute the average and the standard deviation of the refractive index. In Python, this can be done with

```
import numpy as np
ri = np.loadtxt("sphere_edge_projection_statistics.txt", usecols=(1,))
```

(continues on next page)

(continued from previous page)

```
print("average: ", np.average(ri))
print("standard deviation: ", np.std(ri))
```

which will yield a refractive index of 1.357 ± 0.004 which agrees well with the value given in reference [SCG+17] (1.356 ± 0.004); The small difference can be explained by a slightly modified analysis pipeline and originally more strict selection criteria.

4.2 T2: HL60 cell analysis (CLI)

4.2.1 Introduction

HL60 cells in suspension are inhomogeneous, almost-spherical objects. To estimate an average refractive index (RI) of an HL60 cell population, the DryMass command *dm_analyze_sphere* can be used. This tutorial reproduces data presented in figure 5d of reference [MSG+18].

4.2.2 Prerequisites

For this tutorial, you need:

- Python 3.6 or above and DryMass version 0.6.0 or above (see *Installing DryMass*)
- Fiji or Windows Photo Viewer (for data visualization)
- Experimental dataset: DHM_HL60_cells.zip [MSGG19]

4.2.3 Find regions of interest

Note: You can skip this part by copying *roi_slices.txt* from *DHM_HL60_cells.zip* into the *DHM_HL60_cells.zip_dm* folder, running *dm_convert* and manually adding the `[roi]` section to *drymass.cfg* with `enabled = False`.

We proceed slightly different than in *tutorial 1*. Before we use the command *dm_analyze_sphere* to extract the RI values of the HL60 cells, we have to modify our configuration. We start by executing

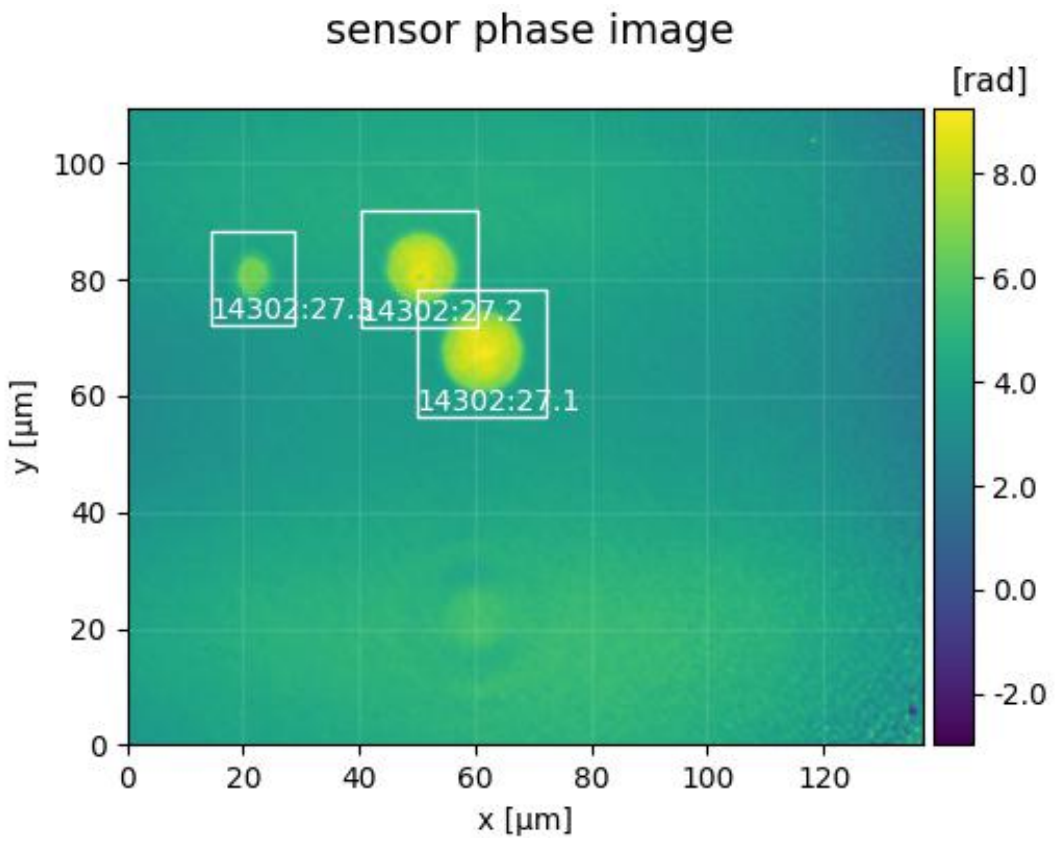
```
dm_extract_roi DHM_HL60_cells.zip
```

which prompts us for the *pixel size* (0.107µm), and the *wavelength* (633nm), which can be found in the *readme.txt* file inside the zip archive. This command imports the raw data and searches for cells in the phase data. Opening the file *sensor_roi_images.tif*, we realize that the search parameters are not set optimally. This is image 27:

We want to exclude small ROIs and ROIs with a large overlap. Furthermore, we want to include all large cells (see e.g. image 39). Thus, we change the following configuration keys in *drymass.cfg*:

```
[specimen]
# approximate cell diameter we are looking for [µm]
size um = 13

[roi]
# increase border size around cells
pad border px = 80
```

(continues on next page)



(continued from previous page)

```
# do not allow large variations of specimen size
size variation = 0.2
# exclude ROIs with an overlap > 100px
exclude overlap px = 100
```

With the new configuration, we run `dm_extract_roi DHM_HL60_cells.zip` again. Now all cells are detected. However, we want to exclude a few due to artifacts or shape issues. To achieve that, we tell DryMass to ignore the corresponding ROIs in *drymass.cfg*

```
[roi]
ignore data = 8.4, 18.2, 18.3, 35.2
```

After executing `dm_extract_roi` again, these ROIs are labeled red in *sensor_roi_images.tif*. There should now be a total of 87 ROIs.

4.2.4 Set 2nd order polynomial background correction

The default setting for background correction in DryMass is *tilt* which means that all phase data are corrected by fitting a 2D tilt image to the image borders. For the present dataset, a second order polynomial fit is a better approach, because the background phase does not follow a linear trend. Thus, we choose the *poly2o* profile and additionally set the fitting border width to 30 pixels. These are the updated lines in the `[bg]` section of *drymass.cfg*:

```
[bg]
phase border px = 30
phase profile = poly2o
```

4.2.5 Perform sphere analysis

We now run

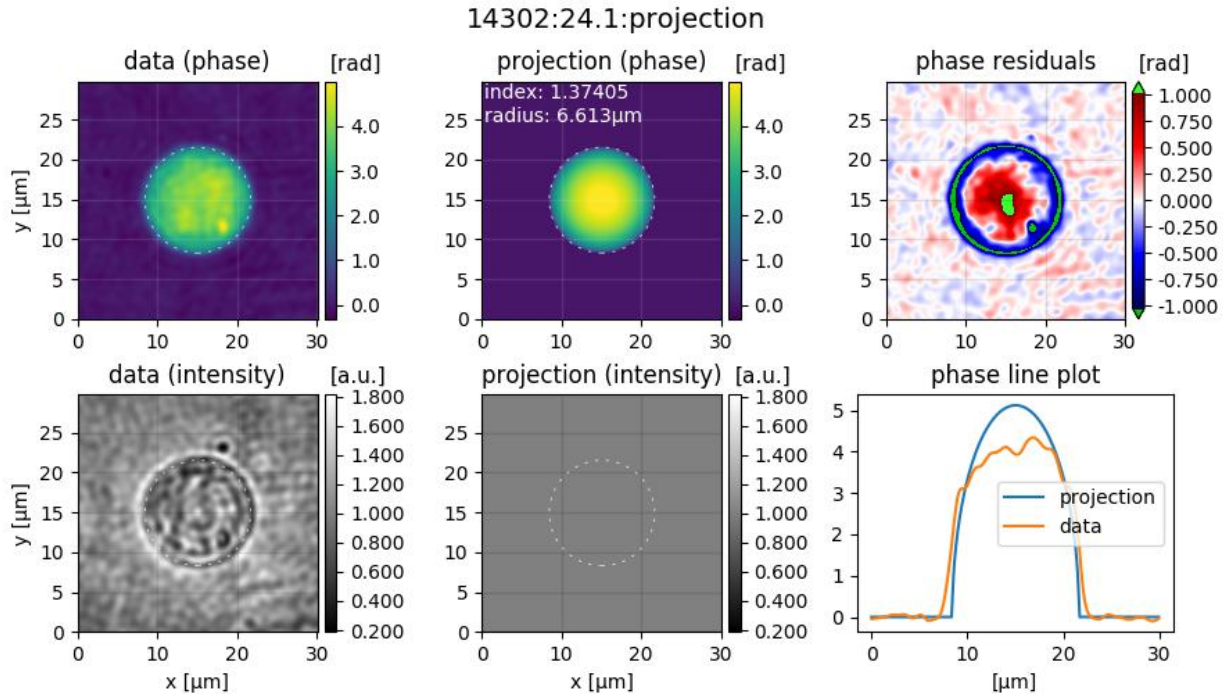
```
dm_analyze_sphere DHM_HL60_cells.zip
```

and are asked to enter the RI of the medium (1.335). By default, the RI of the cells is computed according to [SSM+15]. The following files are created during this step:

- *sphere_edge_projection_data.h5*: QPI data
- *sphere_edge_projection_images.tif*: data visualization
- *sphere_edge_projection_statistics.txt*: results

Note: If you are using *roi_slices.txt* (above note), you can safely ignore warnings about *slice and QPImage identifiers*. Setting the RI of the medium changes the internal ROI identifiers. Since we have fixed the ROIs, the identifiers do not match anymore, but the enumeration is still correct.

Let's have a look at the visualization of ROI 24.1 in *sphere_edge_projection_images.tif*.



The first column shows the experimental data, the second column shows the modeled data (with the cell perimeter indicated by a dashed circle), and the third column contains a residual image (pay attention to the colorbar, green means that the values are outside of the displayed range) and a line plot through the center of the cell. What is most striking about these data is that the RI is overestimated while the radius is underestimated by the edge-projection model. The explanation is that the radius of the cell is determined with an edge-detection algorithm applied to the phase image. Since the edge-detection algorithm determines the edge on the slope of the phase profile and not where the phase profile starts to deviate from the background, it underestimates the radius. The solution to this problem is to take into account the full phase image when determining RI and radius [KKL+07] [MSG+18].

This can be achieved by modifying the `[sphere]` section of *drymass.cfg*. In figure 5d of reference [MSG+18], multiple RI-retrieval methods are applied and compared for the same cell population. To repdroduce these data, we

run `dm_analyze_sphere DHM_HL60_cells.zip` three more times with a modified `[sphere]` section (note that this may take a while).

- Run 1: phase image fit with a projection model

```
[sphere]
method = image
model = projection
```

  which produces the files

  - *sphere_image_projection_data.h5*
  - *sphere_image_projection_images.tif*
  - *sphere_image_projection_statistics.txt*

- Run 2: phase image fit with the Rytov approximation

```
[sphere]
method = image
model = rytov
```

  which produces the files

  - *sphere_image_rytov_data.h5*
  - *sphere_image_rytov_images.tif*
  - *sphere_image_rytov_statistics.txt*

- Run 3: phase image fit with the systematically corrected Rytov approximation

```
[sphere]
method = image
model = rytov-sc
```

  which produces the files

  - *sphere_image_rytov-sc_data.h5*
  - *sphere_image_rytov-sc_images.tif*
  - *sphere_image_rytov-sc_statistics.txt*

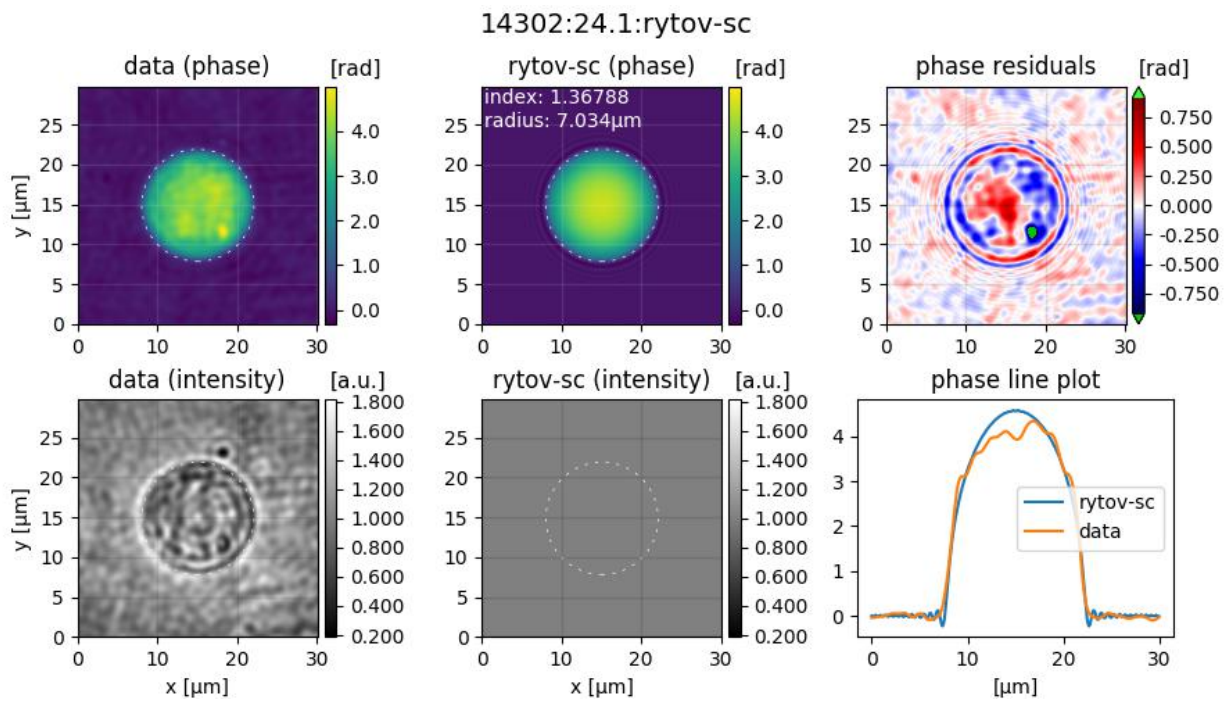
Note: We omitted the case `model = mie-avg` which is part of figure 5d in reference [MSG+18], because of the long fitting time.

To verify that the full-phase-image-based approaches indeed yield lower residuals than the edge-detection approach, let's have a look at ROI 24.1 of *sphere_image_rytov-sc_images.tif*.

The phase difference and the phase line plots look much better now. Observed deviations mostly originate from the inhomogeneity of the cell.

4.2.6 Plot the results

To plot the results, we use the `following Python script`.

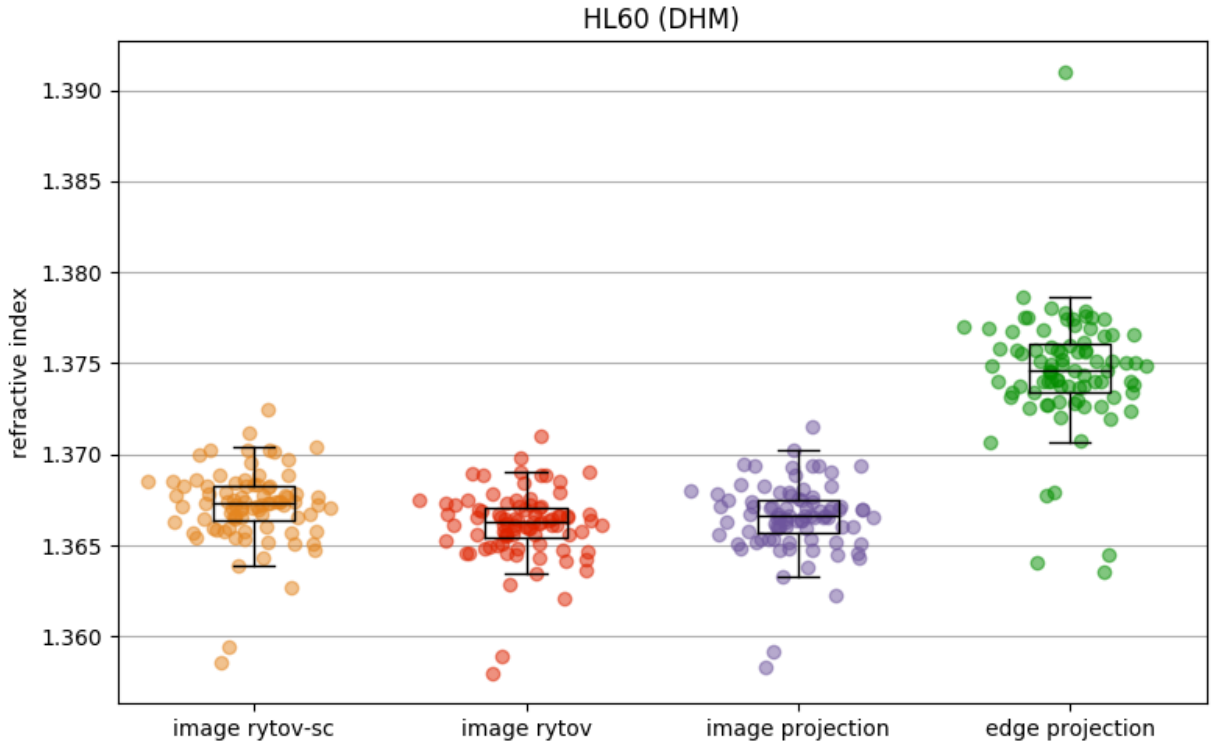



```python
import matplotlib.pylab as plt
import numpy as np


def dot_boxplot(ax, data, colors, labels, **kwargs):
    """Combined box and scatter plot"""
    box_list = []

    for ii in range(len(data)):
        # set same random state for every scatter plot
        rs = np.random.RandomState(42).get_state()
        np.random.set_state(rs)
        y = data[ii]
        x = np.random.normal(ii+1, 0.15, len(y))
        plt.plot(x, y, 'o', alpha=0.5, color=colors[ii])
        box_list.append(y)

    ax.boxplot(box_list,
               sym="",
               medianprops={"color": "black", "linestyle": "solid"},
               widths=0.3,
               labels=labels,
               **kwargs)
    plt.grid(axis="y")


if __name__ == "__main__":
    ri_data = [
        np.loadtxt("sphere_image_rytov-sc_statistics.txt", usecols=(1,)),
        np.loadtxt("sphere_image_rytov_statistics.txt", usecols=(1,)),
        np.loadtxt("sphere_image_projection_statistics.txt", usecols=(1,)),
        np.loadtxt("sphere_edge_projection_statistics.txt", usecols=(1,)),
```

(continues on next page)

(continued from previous page)

```
        ]
    colors = ["#E48620", "#DE2400", "#6e559d", "#048E00"]
    labels = ["image rytov-sc", "image rytov",
              "image projection", "edge projection"]

    plt.figure(figsize=(8, 5))
    ax = plt.subplot(111, title="HL60 (DHM)")
    ax.set_ylabel("refractive index")
    dot_boxplot(ax=ax, data=ri_data, colors=colors, labels=labels)
    plt.tight_layout()
    plt.show()
```



4.2.7 Discussion

The above figure correctly reproduces the message conveyed with figure 5d of reference [MSG+18]. There are only minor differences that can be explained by a slightly different analysis pipeline:

- In [MSG+18], 84 cells were analyzed as opposed to the 87 cells shown here. This can be attributed to the improved object detection pipeline introduced in DryMass 0.1.4.
- In [MSG+18], the phase data were background-corrected with background data (not included in *DHM_HL60_cells.zip*) and a linear model (`phase profile = tilt`) as opposed to a second order polynomial model (which was introduced in DryMass 0.1.3). However, this does not seem to have any significant effect on the results, which indicates that the analysis methods are robust.
- There is a prominent outlier in the *edge projection* results set. The reason for this outlier is a falsely detected contour (see ROI 1.0). This ROI was not included in the analysis of [MSG+18].

- Other minor differences might originate from the fact that the hologram data is processed differently (`[holo]` section of *drymass.cfg*). In [MSG+18], a gaussian filter is used whereas DryMass defaults to a disk filter. For more information on this topic, see e.g. Hologram filter choice.

4.3 T3: Correcting for background and geometry (CLI)

4.3.1 Introduction

In practice, recorded phase data is often subject to a non-linear background phase due to limitations of the measuring setup. Furthermore, the measured objects (e.g. cells, beads, liquid droplets) might be positioned too close to each other or the imaged region. DryMass offers several ways of dealing with the resulting artifacts. This tutorial uses polyacrylamide (PAA) bead measurements that were artificially altered to simulate the following cases:

1. A constant offset in the phase background.
2. A linear tilt in the phase background along one coordinate axis.
3. A linear tilt in the phase background along both coordinate axes.
4. A quadratic phase distortion along one axis and a phase tilt along the other axis.
5. Quadratic phase distortions along both coordinate axes.
6. Two beads that are close to each other.
7. All of the above.
8. A bead that is cut at the image border.

Please note that these are designed examples, i.e. the methods described here to fix these artifacts might not always work and their combination might not work for heterogeneous samples.

4.3.2 Prerequisites

For this tutorial, you need:

- Python 3.6 or above and DryMass version 0.5.0 or above (see *Installing DryMass*)
- Fiji or Windows Photo Viewer (for data visualization)
- Experimental dataset: QLSR_PAA_beads_modified.zip [MSGG19]

4.3.3 Take a glimpse at the data

For this tutorial, the downloaded zip archive has to be extracted prior to the analysis. The extracted archive contains a readme file and the experimental data *QLSR_PAA_beads_modified.h5* in the qpimage file format. DryMass can extract all relevant metadata from this file format (in contrast to the file formats used in tutorials 1 and 2), such that no manual intervention is required when running *dm_analyze_sphere*:

```
dm_analyze_sphere QLSR_PAA_beads_modified.h5
```

Please open the output folder (*QLSR_PAA_beads_modified.h5_dm*) and take a look at the file *sensor_roi_images.tif*. You can see the different artifacts discussed above (offset, tilt, etc.). In the file *sphere_edge_projection_images.tif*, you see that the default background correction strategy works for the offset and tilt artifacts, but fails when the background phase has quadratic components. Furthermore, notice that for the last three measurements no regions of interest (ROIs) were detected with the default parameters. As a result, these three measurements are not represented in *sphere_edge_projection_images.tif*. These issues are discussed and resolved in the following.

4.3.4 Select 2nd order polynomial background correction

A simple tilt-based background correction does not work for the measurements with the quadratic background. Setting the background correction to a second order polynomial resolves this issue. In the configuration file *drymass.cfg* (located in the output folder), edit the *[bg]* section:

```
[bg]
phase profile = poly2o
```

Now the background is sufficiently flat for the first five ROIs.

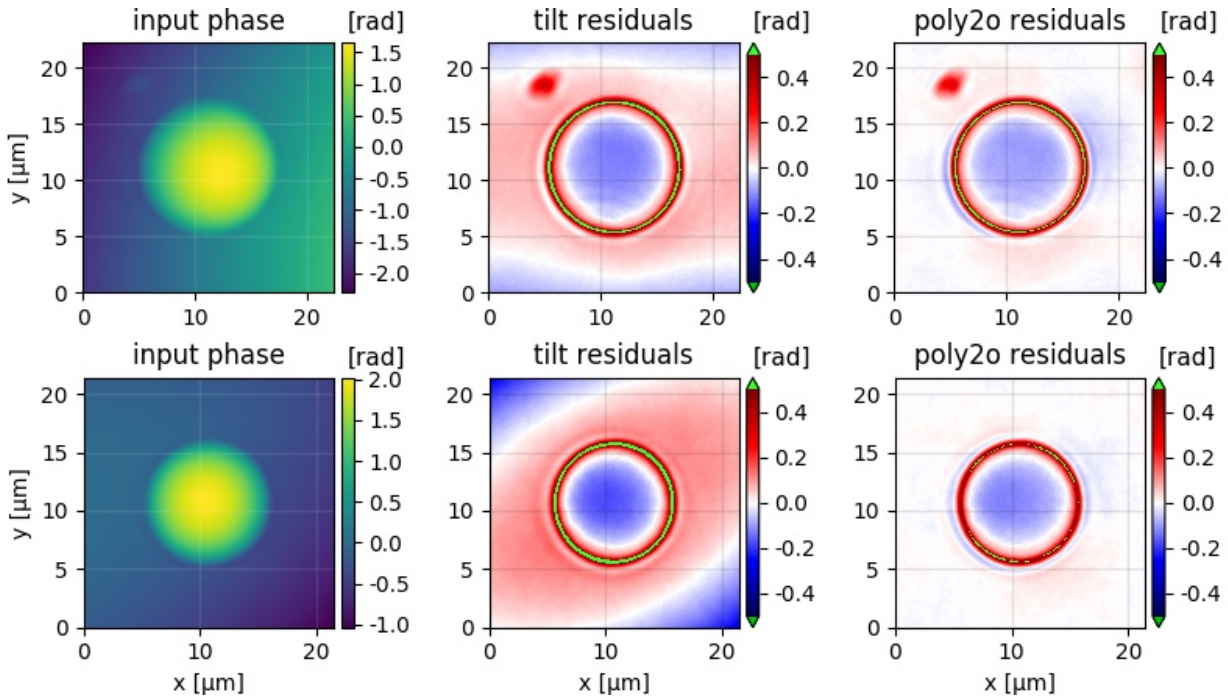


Fig. 1: Comparison of tilt correction and poly2o correction. The first row shows a bead with a quadratic background phase along one axis and a phase tilt along the other. The second row shows a quadratic background phase along both coordinate axes. The first column shows the raw input phase, and the second and third columns show the phase residuals when using the tilt and poly2o correction for a sphere analysis based on the edge-detection algorithm.

4.3.5 Include beads that are close to each other

DryMass automatically removes ROIs that are close to each other, as this might have a negative effect on the subsequent analysis steps. To include these beads, we lift this restriction by modifying the *[roi]* section:

```
[roi]
exclude overlap px = 0
```

In *sensor_roi_images.tif*, you can see that the ROIs of the beads are now included in the analysis. However, in *sphere_edge_projection_images.tif* you observe that the second bead seems to have a negative effect on the background correction. To resolve this issue, we set a binary threshold in the original ROI above which no data is used for background correction. Since it is difficult to set such a threshold manually, we use one of the threshold filters implemented in scikit-image that works well for this example: triangle

```
[bg]
phase binary threshold = triangle
```

Now we have established a robust background correction pipeline that includes all but one bead.

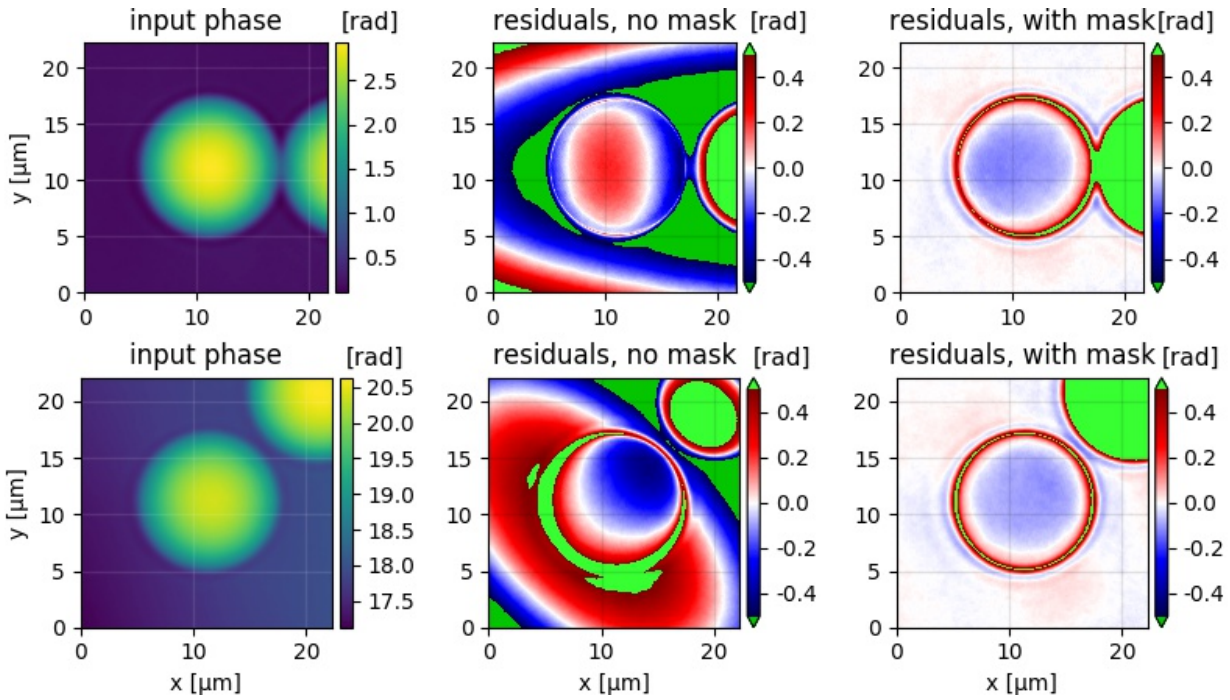


Fig. 2: Phase binarization for background correction. Each row shows one exemplary bead in close proximity to another bead. A simple background correction using the pixels at the border of the image (second column) does not work because of the second bead. To resolve this issue, triangle thresholding is used to use only those pixels for background correction that do not belong to a bead (third column).

4.3.6 Include beads at the border of the sensor image

By default, all ROIs that are within ten pixels of the border of the sensor image are removed from the analysis. We can include all ROIs by setting this distance to zero:

```
[roi]
dist border px = 0
```

The bead in the final measurement is now included in the analysis, yielding values for refractive index and radius.

4.3.7 Exact determination of radius and refractive index

At this point, the tutorial is already complete in the sense that all cases given in the introduction have been covered. However, the residuals of the sphere model are still large, which can be attributed to the default analysis method of *dm_analyze_sphere*: The edge-detection algorithm, as implemented in DryMass, causes an underestimation of the beads radii and thus an overestimation of the refractive index. To retrieve more reliable results, we modify the *[sphere]* section to use the systematically-corrected Rytov approximation (see [MSG+18]):

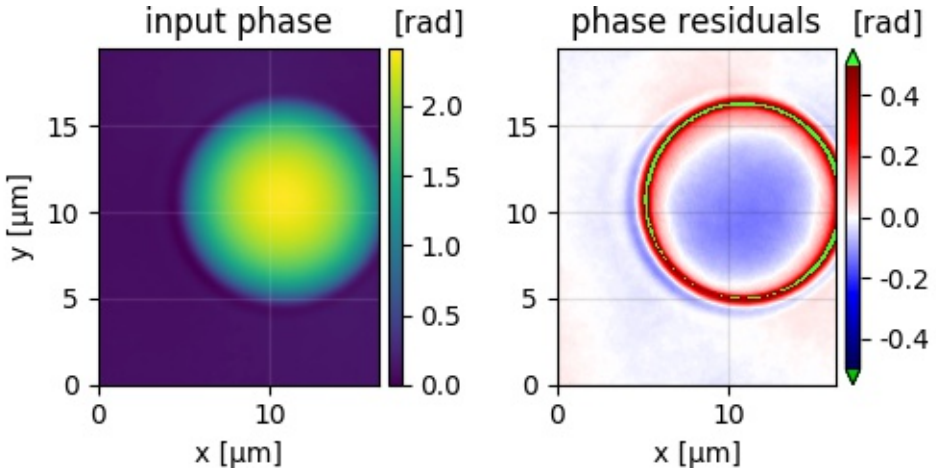


Fig. 3: Objects at the image border can be included in the analysis.

```
[sphere]
method = image
model = rytov-sc
```

In addition to the previously achieved flat phase background for each ROI, this approach minimizes phase residuals and results in more accurate values for refractive index and size of the PAA beads.

4.4 T4: Automating the analysis of large datasets (CLI)

4.4.1 Introduction

This tutorial showcases the command-line parameters `--recursive` and `--profile` (see *Advanced usage*) to automate data analysis, exemplarily the analysis pipeline of *tutorial 2*. After this tutorial, you will be able to automate the application of multiple analysis pipelines with additional, custom analysis steps to large datasets.

4.4.2 Prerequisites

For this tutorial, you need:

- Python 3.6 or above and DryMass version 0.8.0 or above (see *Installing DryMass*)
- Experimental dataset: DHM_HL60_cells.zip [MSGG19]

4.4.3 Setup example measurements

To simulate the presence of several datasets, create a few copies of `DHM_HL60_cells.zip` in a designated folder (here `C:\\Path\to\data\`). You may also create subfolders and put copies of the dataset there. The contents of the folder could be three copies:

- `DHM_HL60_cells_01.zip`
- `DHM_HL60_cells_02.zip`
- `DHM_HL60_cells_03.zip`

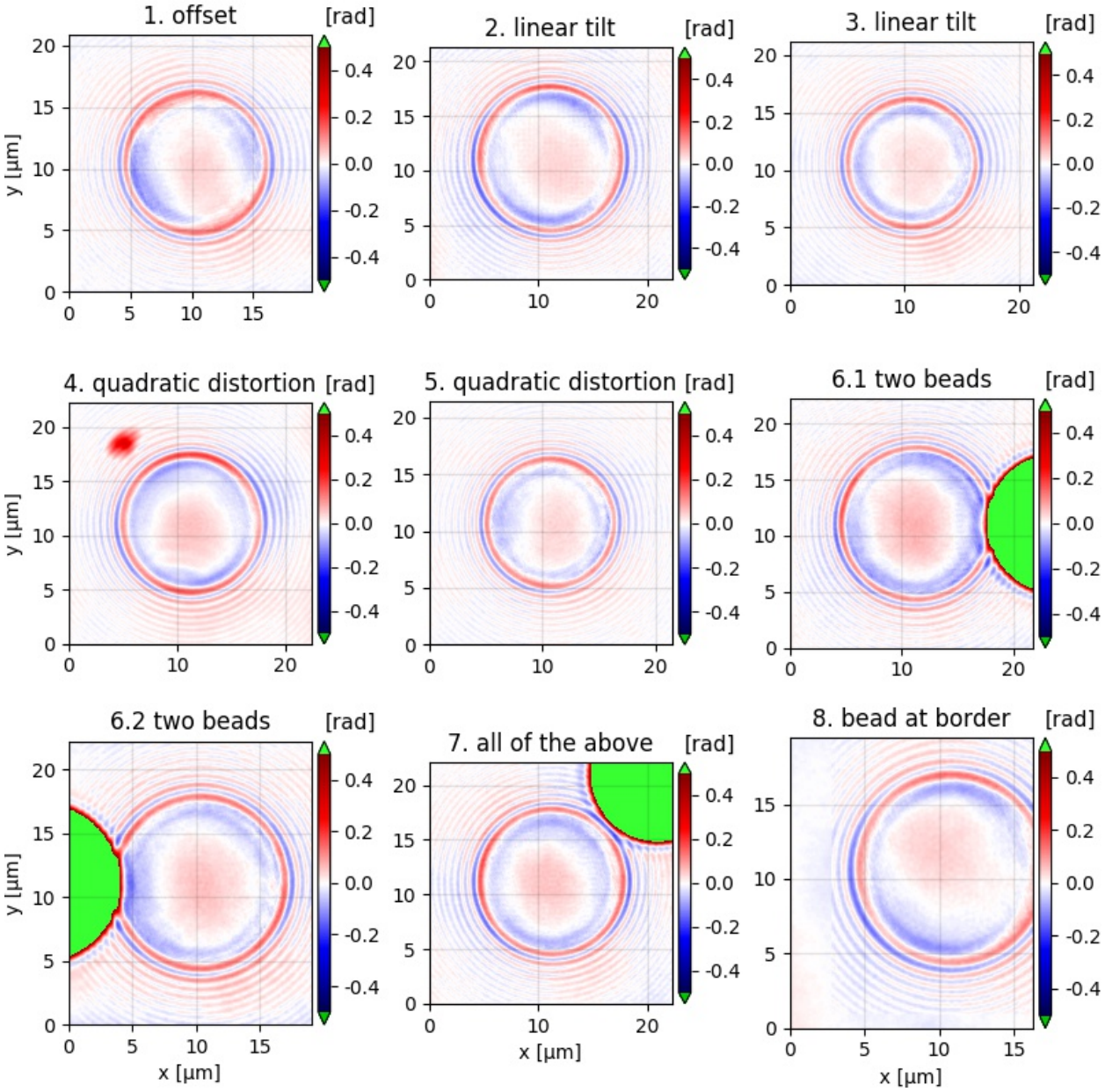


Fig. 4: Phase residuals when fitting with the Rytov approximation. The plots correspond to the different cases presented in the introduction, demonstrating correct background correction and object identification. The residuals are reduced significantly when compared to the edge-detection approach (compare figures above).

4.4.4 Create and import analysis profiles

In *tutorial 2*, the analysis pipeline is executed four times with a different `[sphere]` section. For each of these runs, we create a separate profile and import it into the local library of DryMass so that we may use it later on. The contents of the first profile are:

```
[bg]
phase border px = 30
phase profile = poly2o

[meta]
medium index = 1.335
pixel size um = 0.107
wavelength nm = 633.0

[roi]
pad border px = 80
size variation = 0.2
exclude overlap px = 100
ignore data = 8.4, 15.2, 18.2, 18.3, 35.2

[specimen]
size um = 13

[sphere]
method = edge
model = projection
```

The other profiles are identical to the first profile, except for the `[sphere]` section. First, download all profiles:

- edge-projection: `t04_profile_edge.cfg`
- image-projection: `t04_profile_proj.cfg`
- image-rytov: `t04_profile_rytov.cfg`
- image-rytov-sc: `t04_profile_rytov-sc.cfg`

and then import them into the local library via:

```
dm_profile add t4edge t04_profile_edge.cfg
dm_profile add t4proj t04_profile_proj.cfg
dm_profile add t4rytov t04_profile_rytov.cfg
dm_profile add t4rytov-sc t04_profile_rytov-sc.cfg
```

Once imported in the local library, the downloaded profiles may safely be removed. You can list the available profile with the command `dm_profile list`, which should yield an output similar to this:

```
Available profiles:
 - t4edge: C:\\Users\Something\drymass\profile_t4edge.cfg
 - t4proj: C:\\Users\Something\drymass\profile_t4proj.cfg
 - t4rytov-sc: C:\\Users\Something\drymass\profile_t4rytov-sc.cfg
 - t4rytov: C:\\Users\Something\drymass\profile_t4rytov.cfg
```

4.4.5 Test the analysis pipeline

Before the next level of automation, let us first test the current analysis pipeline:

```
dm_analyze_sphere --recursive --profile t4edge "C:\\Path\to\data\"
```

where `C:\\Path\to\data\` is the folder containing the experimental data which is searched recursively (`--recursive`) and `t4edge` is the name of the profile that employs the edge-detection approach to determine the radius and the refractive index of the cells. Now, verify that all datasets were detected and that the analysis results are identical to those of *tutorial 2*. The output of the above command should be similar to:

```
DryMass version 0.8.0
Recursing into directory tree... Done.
Input 1/3: C:\\Path\to\data\DHM_HL60_cells_01.zip
Input 2/3: C:\\Path\to\data\DHM_HL60_cells_02.zip
Input 3/3: C:\\Path\to\data\DHM_HL60_cells_03.zip
Analyzing dataset 1/3.
Converting input data... Done.
Extracting ROIs... Done.
Plotting detected ROIs... Done
Performing sphere analysis... Done.
Plotting sphere images... Done
Analyzing dataset 2/3.
Converting input data... Done.
Extracting ROIs... Done.
Plotting detected ROIs... Done
Performing sphere analysis... Done.
Plotting sphere images... Done
Analyzing dataset 3/3.
Converting input data... Done.
Extracting ROIs... Done.
Plotting detected ROIs... Done
Performing sphere analysis... Done.
Plotting sphere images... Done
```

4.4.6 Further automation

In principle, we could now run all commands in succession to obtain the fitting results for all model functions:

```
dm_analyze_sphere --recursive --profile t4edge "C:\\Path\to\data\"
dm_analyze_sphere --recursive --profile t4proj "C:\\Path\to\data\"
dm_analyze_sphere --recursive --profile t4rytov "C:\\Path\to\data\"
dm_analyze_sphere --recursive --profile t4rytov-sc "C:\\Path\to\data\"
```

However, since these commands have a comparatively long running time, it makes sense to write a script that can run these commands automatically for a given path.

Windows users can create a command-script, a text file with the `.cmd` extension (e.g. `analysis.cmd`, with the following content:

```
dm_analyze_sphere --recursive --profile t4edge $1
dm_analyze_sphere --recursive --profile t4proj $1
dm_analyze_sphere --recursive --profile t4rytov $1
dm_analyze_sphere --recursive --profile t4rytov-sc $1
```

Linux and MacOS users can create a bash script (`analysis.sh`), with the following content:

```
#!/bin/bash
dm_analyze_sphere --recursive --profile t4edge $1
```

(continues on next page)

(continued from previous page)

```
dm_analyze_sphere --recursive --profile t4proj $1
dm_analyze_sphere --recursive --profile t4rytov $1
dm_analyze_sphere --recursive --profile t4rytov-sc $1
```

To run the full analysis, you now only need to execute a single command:

```
# Windows users:
cd "C:\\Path\to\script"
.\analysis.cmd "C:\\Path\to\data\"

# Linux/MacOS users:
cd "/path/to/script"
bash analysis.sh "C:\\Path\to\data\"
```

4.4.7 Plotting the method comparison automatically

We would also like to automatically plot the comparison between the methods, as in *tutorial 2*. To achieve this, we modify the original python script to accept a path as a command line argument and store the comparison plot as `comparison.png` in the results directories:

```
import pathlib
import sys

import matplotlib.pylab as plt
import numpy as np


def dot_boxplot(ax, data, colors, labels, **kwargs):
    """Combined box and scatter plot"""
    box_list = []

    for ii in range(len(data)):
        # set same random state for every scatter plot
        rs = np.random.RandomState(42).get_state()
        np.random.set_state(rs)
        y = data[ii]
        x = np.random.normal(ii+1, 0.15, len(y))
        plt.plot(x, y, 'o', alpha=0.5, color=colors[ii])
        box_list.append(y)

    ax.boxplot(box_list,
               sym="",
               medianprops={"color": "black", "linestyle": "solid"},
               widths=0.3,
               labels=labels,
               **kwargs)
    plt.grid(axis="y")


def plot_comparison(path):
    """Comparison plot of analysis results"""
    ri_data = [
        np.loadtxt(path / "sphere_image_rytov-sc_statistics.txt",
                   usecols=(1,)),
        np.loadtxt(path / "sphere_image_rytov_statistics.txt",
```

(continues on next page)

(continued from previous page)

```
                    usecols=(1,)),
        np.loadtxt(path / "sphere_image_projection_statistics.txt",
                    usecols=(1,)),
        np.loadtxt(path / "sphere_edge_projection_statistics.txt",
                    usecols=(1,)),
        ]
    colors = ["#E48620", "#DE2400", "#6e559d", "#048E00"]
    labels = ["image rytov-sc", "image rytov",
              "image projection", "edge projection"]
    plt.figure(figsize=(8, 5))
    ax = plt.subplot(111, title="HL60 (DHM)")
    ax.set_ylabel("refractive index")
    dot_boxplot(ax=ax, data=ri_data, colors=colors, labels=labels)
    plt.tight_layout()
    plt.savefig(path / "comparison.png", dpi=300)


if __name__ == "__main__":
    path = pathlib.Path(sys.argv[-1])
    # recursive search for results directories
    for rp in path.rglob("*_dm"):
        if rp.is_dir():
            plot_comparison(path=rp)
```

Put this python script (`t04_method_comparison.py`) into the same folder as your analysis script and add the following line to the the analysis script:

```
python t04_method_comparison.py $1
```

The final analysis script should now look like this:

- Windows: `t04_analysis.cmd`
- Linux/MacOS: `t04_analysis.sh`

This script fully automates the entire analysis from loading raw data to generating a comparison plot.

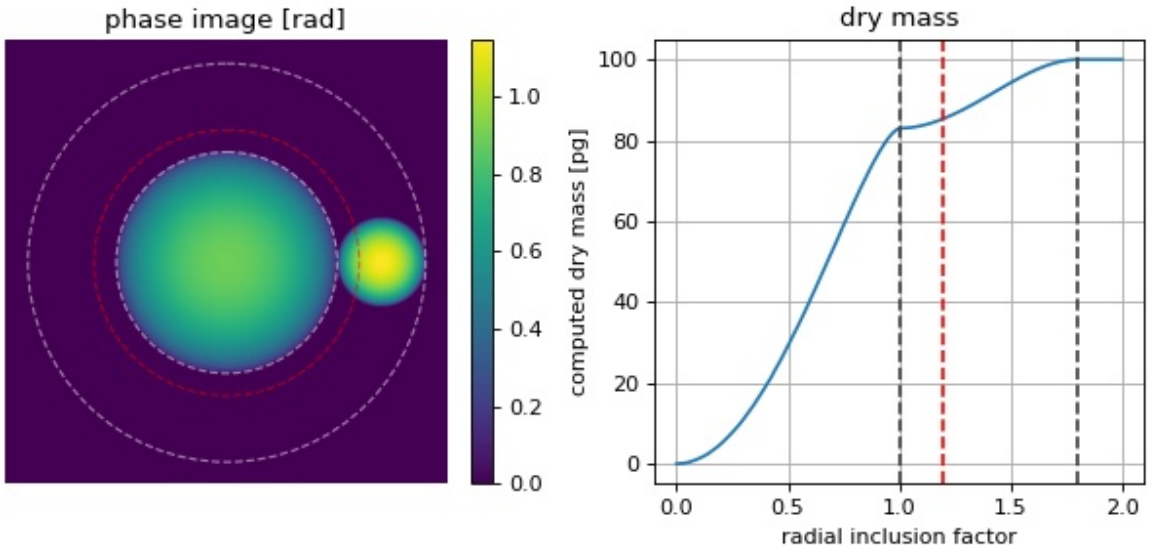
CHAPTER 5

Code examples

5.1 Dry mass computation with radial inclusion factor

This examples illustrates the usage of the "radial inclusion factor" which is defined in the configuration section "sphere" and used in `drymass.anasphere.relative_dry_mass()` with the keyword argument *rad_fact*.

The phase image is computed from two spheres whose dry masses add up to 100pg with the larger sphere having a dry mass of 83pg. The larger sphere is located at the center of the image which is also used as the origin for dry mass computation. The radius of the larger sphere is known (10µm). Thus, the corresponding radius (inner circle) corresponds to a radial inclusion factor of 1. In DryMass, the default radial inclusion factor is set to 1.2 (red). In some cases, this inclusion factor must be increased or decreased depending on whether additional information (the smaller sphere) should be included in the dry mass computation or not.



mass_radial_inclusion_factor.py

```python
from drymass.anasphere import relative_dry_mass
import matplotlib
import matplotlib.pylab as plt
import numpy as np
import qpimage
import qpsphere

# refraction increment
alpha = .18  # [mL/g]

# general simulation parameters
medium_index = 1.333
model = "projection"
wavelength = 500e-9  # [m]
pixel_size = 1e-7  # [m]
grid_size = (400, 400)  # [px]

# sphere parameters
dry_masses = [83, 17]  # [pg]
radii = [10, 4]  # [µm]
centers = [(200, 200), (200, 340)]  # [px]

phase_data = np.zeros(grid_size, dtype=float)
for m, r, c in zip(dry_masses, radii, centers):
    # compute refractive index from dry mass
    r_m = r * 1e-6
    alpha_m3g = alpha * 1e-6
    m_g = m * 1e-12
    n = 1.333 + 3 * alpha_m3g * m_g / (4 * np.pi * (r_m**3))
    # generate example dataset
    qpi = qpsphere.simulate(radius=r_m,
                            sphere_index=n,
                            medium_index=medium_index,
                            wavelength=wavelength,
                            pixel_size=pixel_size,
                            model=model,
                            grid_size=grid_size,
                            center=c)
    phase_data += qpi.pha

qpi_sum = qpimage.QPImage(data=phase_data,
                          which_data="phase",
                          meta_data={"wavelength": wavelength,
                                     "pixel size": pixel_size,
                                     "medium index": medium_index})

# compute dry mass in dependence of radius
mass_evolution = []
mass_radii = []
for rad_fact in np.linspace(0, 2.0, 100):
    dm = relative_dry_mass(qpi=qpi_sum,
                           radius=radii[0] * 1e-6,
                           center=centers[0],
                           alpha=alpha,
                           rad_fact=rad_fact)
    mass_evolution.append(dm * 1e12)
```

(continues on next page)

(continued from previous page)

```
    mass_radii.append(rad_fact)

# plot results
fig = plt.figure(figsize=(8, 3.8))
matplotlib.rcParams["image.interpolation"] = "bicubic"
# phase image
ax1 = plt.subplot(121, title="phase image [rad]")
ax1.axis("off")
map1 = ax1.imshow(qpi_sum.pha)
plt.colorbar(map1, ax=ax1, fraction=.048, pad=0.05)
# dry mass vs. inclusion factor
ax2 = plt.subplot(122, title="dry mass")
ax2.plot(mass_radii, mass_evolution)
ax2.set_ylabel("computed dry mass [pg]")
ax2.set_xlabel("radial inclusion factor")
ax2.grid()
# radius indicators
for r in [100, 180]:
    cx = centers[0][0] + .5
    cy = centers[0][1] + .5
    circle = plt.Circle((cx, cy), r,
                        color='w', fill=False, ls="dashed", lw=1, alpha=.5)
    ax1.add_artist(circle)
    ax2.axvline(r / 100, color="#404040", ls="dashed")
# add default
circle = plt.Circle((cx, cy), 120,
                    color='r', fill=False, ls="dashed", lw=1, alpha=.5)
ax1.add_artist(circle)
ax2.axvline(1.2, color="r", ls="dashed")

plt.tight_layout()
plt.show()
```

5.2 Comparison of relative and absolute dry mass

Relative dry mass is the dry mass computed relative to the surrounding medium. If the refractive index of the surrounding medium does not match that of the intracellular fluid (approximately 1.335), then the relative dry mass underestimates the actual dry mass. For a spherical cell, the absolute (corrected) dry mass can be computed as described in the theory section on *dry mass computation*.

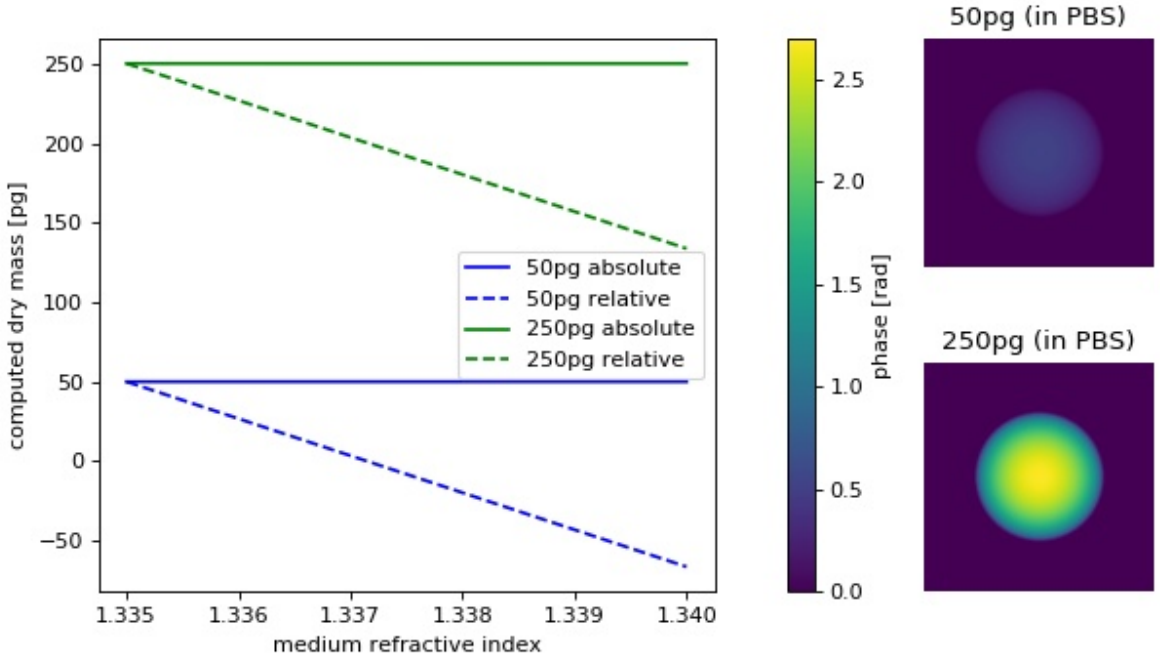
This examples compares the relative dry mass (`drymass.ansphere.relative_dry_mass()`) to the absolute dry mass corrected for a spherical phase object (`drymass.ansphere.absolute_dry_mass_sphere()`). From simulated phase images (projection approach, wavelength 550nm) of two cell-like spheres with a radius of 10µm and dry masses of 50pg (n1.337) and 250pg (n1.346), the absolute and relative dry masses are computed with varying refractive index of the medium.

At the refractive index of phosphate buffered saline (PBS), absolute and relative dry mass are equivalent. As the refractive index of the medium increases, the relative drymass decreases linearly (independent of dry mass), underestimating the actual dry mass.

`mass_relative_vs_absolute.py`

```
from drymass.anasphere import absolute_dry_mass_sphere, relative_dry_mass
import matplotlib
```

(continues on next page)



(continued from previous page)

```
import matplotlib.pylab as plt
import numpy as np
import qpsphere

# refraction increment
alpha = .18  # [mL/g]

# general simulation parameters
model = "projection"
wavelength = 500e-9  # [m]
pixel_size = 1.8e-7  # [m]
grid_size = (200, 200)  # [px]

# sphere parameters
radius = 10  # [µm]
center = (100, 100)  # [px]

dry_masses = [50, 250]  # [pg]
medium_indices = np.linspace(1.335, 1.34, 5)

qpi_pbs = {}
m_abs = {}
m_rel = {}
phase_data = np.zeros(grid_size, dtype=float)
for m in dry_masses:
    # initiate results list
    m_abs[m] = []
    m_rel[m] = []
    # compute refractive index from dry mass
    r_m = radius * 1e-6
    alpha_m3g = alpha * 1e-6
```

(continues on next page)

(continued from previous page)

```
    m_g = m * 1e-12
    n = 1.335 + 3 * alpha_m3g * m_g / (4 * np.pi * (r_m**3))
    for medium_index in medium_indices:
        # generate example dataset
        qpi = qpsphere.simulate(radius=r_m,
                                sphere_index=n,
                                medium_index=medium_index,
                                wavelength=wavelength,
                                pixel_size=pixel_size,
                                model=model,
                                grid_size=grid_size,
                                center=center)
        # absolute dry mass
        ma = absolute_dry_mass_sphere(qpi=qpi,
                                      radius=r_m,
                                      center=center,
                                      alpha=alpha,
                                      rad_fact=1.2)
        m_abs[m].append(ma * 1e12)
        # relative dry mass
        mr = relative_dry_mass(qpi=qpi,
                               radius=r_m,
                               center=center,
                               alpha=alpha,
                               rad_fact=1.2)
        m_rel[m].append(mr * 1e12)
        if medium_index == 1.335:
            qpi_pbs[m] = qpi

# plot results
fig = plt.figure(figsize=(8, 4.5))
matplotlib.rcParams["image.interpolation"] = "bicubic"
# phase images
kw = {"vmax": qpi_pbs[dry_masses[1]].pha.max(),
      "vmin": qpi_pbs[dry_masses[1]].pha.min()}

ax1 = plt.subplot2grid((2, 3), (0, 2))
ax1.set_title("{}pg (in PBS)".format(dry_masses[0]))
ax1.axis("off")
map1 = ax1.imshow(qpi_pbs[dry_masses[0]].pha, **kw)

ax2 = plt.subplot2grid((2, 3), (1, 2))
ax2.set_title("{}pg (in PBS)".format(dry_masses[1]))
ax2.axis("off")
ax2.imshow(qpi_pbs[dry_masses[1]].pha, **kw)

# overview plot
ax3 = plt.subplot2grid((2, 3), (0, 0), colspan=2, rowspan=2)
ax3.set_xlabel("medium refractive index")
ax3.set_ylabel("computed dry mass [pg]")
for m, c in zip(dry_masses, ["blue", "green"]):
    ax3.plot(medium_indices, m_abs[m], ls="solid", color=c,
             label="{}pg absolute".format(m))
    ax3.plot(medium_indices, m_rel[m], ls="dashed", color=c,
             label="{}pg relative".format(m))
ax3.legend()
plt.colorbar(map1, ax=ax3, fraction=.048, pad=0.1,
```

(continues on next page)

(continued from previous page)

```
91              label="phase [rad]")
92
93 plt.tight_layout()
94 plt.subplots_adjust(wspace=.14)
95 plt.show()
```

CHAPTER 6

Code reference

6.1 Command-line interface

The usage of the command line interface is described in detail *here*. It consists of these main methods:

6.1.1 cli.config

drymass.cli.config.**FILE_CONFIG = 'drymass.cfg'**
DryMass configuration file name

class drymass.cli.config.**ConfigFile**(*path*)
DryMass configuration file management

Manage configuration file of an experimental data set with restrictions imposed by *drymass.cli.definitions.config*.

Parameters **path** (*str*) – path to the configuration file or a folder containing the configuration file *FILE_CONFIG*.

remove_section(*section*)
Remove a section from the configuration file

set_value(*section*, *key*, *value*)
Set a configuration key value

Parameters

- **section** (*str*) – the configuration section
- **key** (*str*) – the configuration key in *section*
- **value** – the configuration key value

Notes

Valid section and key names are defined in definitions.py

update(*other*)
Update the current configuration with data from another

Parameters **other** (ConfigFile) – the configuration file from which data is imported into the current configuration

Notes

None-valued keys are ignored.

6.1.2 cli.definitions

drymass.cli.definitions.**config**
The keys and subkeys of the definition dictionary are defined and described in the *configuration file section*.

6.1.3 cli.dialog

The dialog submodule contains methods for user-interaction.

drymass.cli.dialog.**OUTPUT_SUFFIX = '_dm'**
DryMass analysis output suffix (appended to data path)

drymass.cli.dialog.**input_setting**
Ask the user for a configuration key

drymass.cli.dialog.**main**(*path=None*, *req_meta=[]*, *description='DryMass analysis.'*, *profile=None*, *recursive=False*)
Main user dialog with optional "meta" kwargs required

Parameters

- **path** (*str, pathlib.Path, or None*) – Path to the measurement data. If set to *None*, the command-line will be parsed.
- **req_meta** (*list of str*) – Keyword arguments of the [meta] section in drymass.cfg that are required by the current task.
- **description** (*str*) – Description of the current task. The description is displayed when the user executes a console_script entry-point with the *–help* argument.
- **profile** (*str, pathlib.Path, or None*) – A path to a 'drymass.cfg' file or a name of a profile in the local library (see drymass.cli.profile). If set to *None*, the default profile is used and the user is asked for missing values.
- **recursive** (*bool*) – Perform recursive search in *path*. If *path* is None, then *recursive* must be False. Instead, the *recursive* argument should be set via the command line.

Returns

- **path_in** (*pathlib.Path or list of pathlib.Path*) – The measurement path. If a recursive search is performed (see *recursive* parameter above), then a list of the measurement paths is returned.
- **path_out** (*pathlib.Path or None*) – The output path, i.e. the path with *_dm* appended. If a recursive search is performed, *path_out* is set to None.

drymass.cli.dialog.**parse**
Obtain the input data set path by parsing the command line

drymass.cli.dialog.**recursive_search**(*path*)
Perform recursive search for supported measurements

drymass.cli.dialog.**transfer_meta_data**(*path_in*, *path_out*)
Read input meta data and write it to the configuration file

6.1.4 cli.parse_funcs

These methods are used to parse the values set in the *Configuration file* and convert them to the correct type.

drymass.cli.parse_funcs.**fbool**(*value*)
Boolean value from string or number

drymass.cli.parse_funcs.**fintlist**(*alist*)
List of integers from string or list of strings/integers

drymass.cli.parse_funcs.**float01**(*flt*)
Float value between 0 and 1

drymass.cli.parse_funcs.**float_or_str**(*flt_or_str*)
Float value from string or number

drymass.cli.parse_funcs.**floattuple_or_one**(*fti*)
Tuple of two floats or ± 1 from a string or a number

drymass.cli.parse_funcs.**int_or_path**(*intpath*)
Integer or string from a string or a number

drymass.cli.parse_funcs.**lcstr**(*astr*)
Convert a string to lower-case

drymass.cli.parse_funcs.**strlist**(*alist*)
List of strings, comma- or space-separated

drymass.cli.parse_funcs.**strlist_vsort**(*alist*)
Same as *strlist* except sorted according to version-representation

drymass.cli.parse_funcs.**tupletupleint**(*items*)
A tuple containing x- and y- slice tuples from a string or tuple

6.1.5 cli.plot

DryMass plotting functionalities.

drymass.cli.plot.**add_cbar**(*ax*, *mapper*, *fmt='%.2f'*, *units=''*, *loc='right'*, *size='5%'*, *labelloc=None*, *extend='neither'*)
Add a colorbar to a plot

drymass.cli.plot.**plot_image**(*data*, *ax=None*, *imtype='phase'*, *cbar=True*, *px_um=None*, *ret_cbar=False*, *cbformat=None*, ***kwargs*)
Plot an image

Parameters

- **data** (*2d np.ndarray*) – Input image
- **ax** (*matplotlib.Axes*) – Axis to plot to
- **imtype** (*str*) – One of [“intensity”, “phase”, “phase error”, “refractive index”].

- **cbar** (*bool*) – Whether to add a colorbar.
- **px_um** (*float*) – Pixel size [µm]
- **ret_cbar** (*bool*) – Whether to return the colorbar.
- **kwargs** (*dict*) – Keyword arguments to *plt.imshow*.

Returns Axis and colorbar.

Return type ax [, cbar]

drymass.cli.plot.**plot_qpi_phase**(*qpi*, *rois=None*, *path=None*, *labels_excluded=[]*)
Plot phase data

drymass.cli.plot.**plot_qpi_sphere**(*qpi_real*, *qpi_sim*, *path=None*, *simtype='simulation'*)
Plot QPI sphere analysis data

6.2 Data analysis

6.2.1 anasphere

drymass.anasphere.**FILE_SPHERE_DATA = 'sphere_{}_{}_data.h5'**
Output sphere analysis qpimage.QPSeries data

drymass.anasphere.**FILE_SPHERE_STAT = 'sphere_{}_{}_statistics.txt'**
Output sphere analysis statistics

exception drymass.anasphere.**EdgeDetectionFailedWarning**

drymass.anasphere.**analyze_sphere**(*h5roi*, *dir_out*, *r0=1e-05*, *method='edge'*, *model='projection'*, *edgekw={}*, *imagekw={}*, *alpha=0.18*, *rad_fact=1.2*, *ret_changed=False*, *ret_reused=False*, *count=None*, *max_count=None*)
Perform sphere analysis

Parameters

- **h5series** (*str*) – Path of qpimage.QPSeries hdf5 file
- **dir_out** (*str*) – Path to output directory
- **r0** (*float*) – Initial radius
- **method** (*str*) – Either "edge" or "image"; see *[sphere] Sphere-based image analysis* for more information.
- **model** (*str*) – Propagation model to use; see *[sphere] Sphere-based image analysis* for more information.
- **edgekw** (*dict*) – Keyword arguments to `qpsphere.edgefit.contour_canny()`
- **imagekw** (*dict*) – Keyword arguments to `qpsphere.imagefit.alg.match_phase()`
- **alpha** (*float*) – Refraction increment [mL/g]
- **rad_fact** (*float*) – Radial inclusion factor for dry mass computation
- **ret_changed** (*bool*) – Return boolean indicating whether the sphere data on disk was created/updated (True) or whether only previously created ROI data was used (False).
- **ret_reused** (*bool*) – Return integer indicating how many previous fits were reused.

- **max_count** (*count,*) – Can be used to monitor the progress of the algorithm. Initially, the value of *max_count.value* is incremented by the total number of steps. At each step, the value of *count.value* is incremented.

drymass.anasphere.**absolute_dry_mass_sphere**(*qpi*, *radius*, *center*, *alpha=0.18*, *rad_fact=1.2*)

Compute absolute dry mass of a spherical phase object

The absolute dry mass is computed with

```
m_abs = m_rel + m_sup
m_rel = lambda / (2*PI*alpha) * phi_tot * deltaA
m_sup = 4*PI / (3*alpha) * radius^3 (n_med - n_PBS)
```

with the vacuum wavelength `lambda`, the total phase retardation in the area of interest `phi_tot`, the pixel area `deltaA`, the refractive index of the medium *n_med* (stored in `qpi.meta`), and the refractive index of phosphate buffered saline (PBS) `n_PBS=1.335`.

This is the *absolute* dry mass, because it takes into account the offset caused by the suppressed density in the phase data.

Parameters

- **qpi** (*qpimage.QPImage*) – QPI data
- **center** (*tuble (x,y)*) – Center of the sphere [px]
- **radius** (*float*) – Radius of the sphere [m]
- **wavelength** (*float*) – The wavelength of the light [m]
- **alpha** (*float*) – Refraction increment [mL/g]
- **rad_fact** (*float*) – Inclusion factor that scales *radius* to increase the area used for phase summation; if the backgound phase exhibits a noise phase signal, positive and negative contributions cancel out and *rad_fact* does not have an effect above a certain critical value.

Returns **dry_mass** – The absolute dry mass of the sphere [g]

Return type float

drymass.anasphere.**relative_dry_mass**(*qpi*, *radius*, *center*, *alpha=0.18*, *rad_fact=1.2*)

Compute relative dry mass of a circular area in QPI

The dry mass is computed with

m_rel = lambda / (2*PI*alpha) * phi_tot * deltaA

with the vacuum wavelength *lambda*, the total phase retardation in the area of interest *phi_tot*, and the pixel area *deltaA*.

This is the *relative* dry mass, because it is computed relative to the surrounding medium (phi_tot) and not relative to water.

Parameters

- **qpi** (*qpimage.QPImage*) – QPI data
- **center** (*tuble (x,y)*) – Center of the area of interest [px]
- **radius** (*float*) – Radius of the area of interest [m]
- **wavelength** (*float*) – The wavelength of the light [m]
- **alpha** (*float*) – Refraction increment [mL/g]

- **rad_fact** (*float*) – Inclusion factor that scales *radius* to increase the area used for phase summation; if the backgound phase exhibits a noise phase signal, positive and negative contributions cancel out and *rad_fact* does not have an effect above a certain critical value.

Returns **dry_mass** – The relative dry mass of the object [g]

Return type float

6.2.2 converter

drymass.converter.**FILE_SENSOR_DATA_H5 = 'sensor_data.h5'**
Output qpimage.QPSeries sensor data

drymass.converter.**FILE_SENSOR_DATA_TIF = 'sensor_data.tif'**
Output phase/amplitude TIFF sensor data

drymass.converter.**convert**(*path_in*, *dir_out*, *meta_data={}*, *holo_kw={}*, *bg_data_amp=None*, *bg_data_pha=None*, *write_tif=False*, *ret_dataset=False*, *ret_changed=False*, *count=None*, *max_count=None*)
Convert experimental data to *qpimage.QPSeries* on disk

Parameters

- **path_in** (*str or pathlib.Path*) – Input path to file or directory
- **dir_out** (*str or pathlib.Path*) – Outuput direcory
- **meta_data** (*dict*) – Meta data (see *qpimage.meta.DATA_KEYS*)
- **bg_data_pha** (*bg_data_amp,*) – The background data for phase and amplitude. One of
  - *None*: No background data
  - *int*: Image index (starting at 1) of the input data set to use as background data
  - *str*, *pathlib.Path*: Path to a separate file that is used for background correction, relative to the directory in which *path_in* is located (*path_in.parent*).
- **write_tif** (*bool*) – Export tif images for use with Fiji/ImageJ (tif images are only created if they don't already exist or if the analysis changed)
- **ret_dataset** (*bool*) – Return the qpformat dataset
- **ret_changed** (*bool*) – Return True if the dataset changed
- **max_count** (*count,*) – Can be used to monitor the progress of the algorithm. Initially, the value of *max_count.value* is incremented by the total number of steps. At each step, the value of *count.value* is incremented.

drymass.converter.**get_background**(*bg_data*, *dataset*, *which='phase'*)
Obtain the background data for a dataset

Parameters

- **bg_data** (*None, int, str, or pathlib.Path*) – Represents the background data:
  - None: no background data
  - int: image with index *bg_data - 1* in *dataset* is used for background correction
  - str, pathlib.Path: An external file will be used for background correction.

- **dataset** (*qpformat.dataset.SeriesData*) – The dataset for which the background data is collected. No background correction is performed! *dataset* is needed for integer *bg_data* and for path-based *bg_data* (because of meta data and hologram kwargs).

Returns **bg** – The background data.

Return type 2d np.ndarray

drymass.converter.**h5series2tif**(*h5in*, *tifout*)
Convert a qpimage.QPSeries file to a phase/amplitude TIFF file

6.2.3 extractroi

drymass.extractroi.**BG_DEFAULT_KW = {'border_perc': 5, 'border_px': 5, 'fit_offset': 'mea**
Default background correction keyword arguments

drymass.extractroi.**FILE_ROI_DATA_H5 = 'roi_data.h5'**
Output ROI qpimage.QPSeries data

drymass.extractroi.**FILE_ROI_DATA_TIF = 'roi_data.tif'**
Output phase/amplitude TIFF data

drymass.extractroi.**FILE_SLICES = 'roi_slices.txt'**
Output slice locations

drymass.extractroi.**extract_roi**(*h5series*, *dir_out*, *size_m*, *size_var=0.5*, *max_ecc=0.7*, *dist_border=10*, *pad_border=40*, *exclude_overlap=30.0*, *threshold='li'*, *ignore_data=None*, *force_roi=None*, *bg_amp_kw={'border_perc': 5, 'border_px': 5, 'fit_offset': 'mean', 'fit_profile': 'tilt'}*, *bg_amp_bin=None*, *bg_amp_mask_radial_clearance=None*, *bg_pha_kw={'border_perc': 5, 'border_px': 5, 'fit_offset': 'mean', 'fit_profile': 'tilt'}*, *bg_pha_bin=None*, *bg_pha_mask_radial_clearance=None*, *bg_sphere_edge_kw={}*, *search_enabled=True*, *ret_roimgr=False*, *ret_changed=False*, *count=None*, *max_count=None*)
Extract ROIs from a qpimage.QPSeries hdf5 file

Parameters

- **h5series** (*str*) – Path of qpimage.QPSeries hdf5 file
- **dir_out** (*str*) – Path to output directory
- **size_m** (*float*) – Approximate diameter of the specimen [m]
- **size_var** (*float*) – Allowed variation relative to specimen size
- **max_ecc** (*float*) – Allowed maximal eccentricity of the specimen
- **dist_border** (*int*) – Minimum distance of objects to image border [px]
- **pad_border** (*int*) – Padding of object regions [px]
- **exclude_overlap** (*float*) – Allowed distance between two objects [px]
- **threshold** (*float or str*) – Thresholding value or method used; see drymass.search.available_thresholds

- **ignore_data** (*list of str*) – Identifiers for sensor images or ROIs to be excluded from further analysis. These will be labeled in the output tiff file and not written to the output qpseries file.
- **force_roi** (*tuple*) – A tuple describing the slice of the ROI to be extracted ((x1, x2), (y1, y2)). This option invalidates all other keyword arguments. If set to *None* (default) an automated search for ROIs is performed.
- **bg_amp_kw** (*dict or None*) – Amplitude image background correction keyword arguments (see `qpimage.QPImage.compute_bg()`), defaults to *BG_DEFAULT_KW*, set to *None* to disable correction
- **bg_amp_bin** (*float, str, or None*) – The amplitude binary threshold value or the method for binary threshold determination; see `skimage.filters` *threshold_** methods. Disabled if set to *None*.
- **bg_amp_mask_radial_clearance** (*float or None*) – If not NaN, use `qpsphere.cnvnc.bg_phase_mask_for_qpi()` to compute a mask image and use it for amplitude background correction. Disabled if set to *None*.
- **bg_pha_kw** (*dict or None*) – Phase image background correction keyword arguments (see `qpimage.QPImage.compute_bg()`), defaults to *BG_DEFAULT_KW*, set to *None* to disable correction
- **bg_pha_bin** (*float, str, or None*) – The phase binary threshold value or the method for binary threshold determination; see `skimage.filters` *threshold_** methods. Disabled if set to *None*.
- **bg_pha_mask_radial_clearance** (*float or None*) – If not NaN, use `qpsphere.cnvnc.bg_phase_mask_for_qpi()` to compute a mask image and use it for phase background correction. Disabled if set to *None*.
- **search_enabled** (*bool*) – If True, perform automated search for ROIs using the parameters above. If False, extract the ROIs from *FILE_SLICES* and only perform background correction using the *bg_** parameters.
- **ret_roimgr** (*bool*) – Return the ROIManager instance of the found ROIs
- **ret_changed** (*bool*) – Return boolean indicating whether the ROI data on disk was created/updated (True) or whether previously created ROI data was used (False).
- **max_count** (*count,*) – Can be used to monitor the progress of the algorithm. Initially, the value of *max_count.value* is incremented by the total number of steps. At each step, the value of *count.value* is incremented.

Notes

The output hdf5 file *dir_out/FILE_ROI_DATA_H5* is a `qpimage.QPSeries` file with the keyword "identifier" consisting of three hashes in the form "hash_data:hash_roiparms:hash_roisexcl", where "hash_data" is the hash of the source dataset, "hash_roiparms", is the hash of the ROI extraction configuration, and "hash_roisexcl" is the hash of the ROI indices excluded.

`drymass.extractroi.`**is_ignored_roi**(*roi*, *ignore_data*)

Determine whether a specific ROI should be ignored

Parameters

- **roi** (*drymass.roi.ROI*) – ROI instance

- **ignore_data** (*list of str*) – List of strings of the form "image_index" or "image_index.roi_index" that identify ROIs that should be ignored. For instance ["1.0", "2.1", "3"].

6.3 Helper classes and methods

6.3.1 search

drymass.search.**approx_bg**(*data*, *filter_size=None*)

Approximate the image background with Gaussian convolution

Parameters

- **data** (*2d ndarray*) – Data from which to compute the background
- **size** (*float*) – Approximate size of the objects on the background. The size of the Gaussian is heuristically determined with

$$\sigma = 5 \cdot \mathtt{size}$$

Returns

Return type Approximate background of *data*.

drymass.search.**search_objects_base**(*image*, *size=110*, *size_var=0.5*, *max_ecc=0.7*, *dist_border=10*, *threshold='li'*, *verbose=False*)

Search objects in images

The wrapper *search_phase_objects()* implements a more robust (heuristic) way of finding objects.

Parameters

- **image** (*2d ndarray*) – Input image
- **size** (*float*) – Approximate diameter of phase objects in pixels
- **size_var** (*float*) – Allowed variation in size (relative to *size*) for the detected objects
- **max_ecc** (*float in interval [0,1)*) – Maximal eccentricity of the objects. The eccentricity of a circle is zero. For an ellipse it is defined as

$$\varepsilon = \sqrt{\frac{a^2 - b^2}{a^2}} = \sqrt{1 - \left(\frac{b}{a}\right)^2} = f/a.$$

- **dist_border** (*float*) – Minimum distance of detected regions to the borders of the image in pixels
- **threshold** (*float or str*) – Thresholding value or method used; see *drymass.threshold.available_thresholds*
- **verbose** (*bool*) – If *True*, print information about ignored regions

Returns **rois** – Found regions

Return type list of regionprops

See also:

skimage.filters.threshold_otsu() threshold for finding objects

skimage.segmentation.clear_border() remove regions at the border

skimage.morphology.label() identify regions in binary images

drymass.search.**search_phase_objects**(*qpi*, *size_m*, *size_var=0.5*, *max_ecc=0.7*, *dist_border=10*, *pad_border=40*, *exclude_overlap=30.0*, *threshold='li'*, *verbose=False*)

Search phase objects in quantitative phase images

Parameters

- **qpi** (*qpimage.QPImage*) – Quantitative phase data
- **size_m** (*float*) – Expected size of the phase objects
- **size_var** (*float*) – Allowed variation in size (relative to *size*) for the detected objects
- **max_ecc** (*float in interval [0,1)*) – Maximal eccentricity of the objects. The eccentricity of a circle is zero. For an ellipse it is defined as

$$\varepsilon = \sqrt{\frac{a^2 - b^2}{a^2}} = \sqrt{1 - \left(\frac{b}{a}\right)^2} = f/a.$$

- **dist_border** (*int*) – Minimum distance of detected regions to the borders of the image in pixels
- **pad_border** (*int*) – Pad the regions of all objects
- **exclude_overlap** (*float*) – Allowed distance in pixels between two detected regions (without *pad_border*)
- **threshold** (*float or str*) – Thresholding value or method used; see *drymass.threshold.available_thresholds*
- **verbose** (*bool*) – If *True*, print information about ignored regions

Returns slices

Return type list of slice

Notes

Description of the heuristic search algorithm:

1. Search phase objects in *qpi.raw_pha* with a background correction computed from the gaussian-filtered image
2. If no objects were found and *qpi.bg_pha* is nonzero, search objects in *qpi.pha*.
3. Search objects in *qpi.bg_pha* and remove any overlaps with the previously obtained regions.

See also:

search_objects_base() underlying search algorithm

approx_bg() gaussian-filtered background estimation

6.3.2 threshold

There are two types of thresholding done in DryMass.

1. Thresholding of the *sensor image* which is required by the detection of the phase object ROIs (`[roi]: threshold` configuration parameter)
2. Thresholding of the *individual ROIs* for determining the masked area used for ROI background correction (`[bg]: amplitude binary threshold` and `[bg]: phase binary threshold` configuration parameters)

drymass.threshold.**image2mask**(*image*, *value_or_method*, *invert=False*)
Convert an image to a binary mask for background correction

If *invert* is False, the threshold value is included in the resulting array.

Parameters

- **image** (*2d np.ndarray*) – Input image
- **value_or_method** (*float or str*) – Either a threshold value or a string naming a filter method in `skimage.filters`.
- **invert** (*bool*) – Invert the resulting boolean array

drymass.threshold.**threshold_drymass_nuclei**(*image*)
Threshold filter for segmenting cell nuclei in phase images

Cell nuclei have a low refractive index, but the nucleoli within the nuclei usually have a very high refractive index. As a result, conventional thresholding algorithms either cannot detect the nuclei reliably or segment the nucleoli only.

This thresholding filter copes with the situation by "pulling down" the top 1% of the phase data and taking the threshold at 20% of the maximum phase relative to the mean of the original phase data.

drymass.threshold.**threshold_li**(*image*)
Li threshold optimized for cells in QPI

drymass.threshold.**threshold_dict = {'dm-nuclei': <function threshold_drymass_nuclei>, 'iso**
Dictionary containing all thresholding methods available in DryMass

drymass.threshold.**available_thresholds = ['dm-nuclei', 'isodata', 'li', 'mean', 'minimum',**
Available thresholding method names; The thresholding methods are either defined in this module (see *threshold_** methods) or taken from `skimage.filters`.

6.3.3 util

Utility methods

drymass.util.**hash_file**(*path*, *blocksize=65536*)
Compute sha256 hex-hash of a file

Parameters

- **path** (*str*) – path to the file
- **blocksize** (*int*) – block size read from the file

Returns **hex** – The first six characters of the hash

Return type str

drymass.util.**hash_object**(*obj*)
Compute sha256 hex-hash of a Python object

Objects in dicts/lists/tuples are joined together before hashing using *obj2bytes()* in this module.

Returns **hex** – The first six characters of the hash

Return type str

drymass.util.**is_series_file**(*path*)
Return True if *path* is a qpimage.QPSeries file with identifier

drymass.util.**obj2bytes**(*obj*)
String representation of an object for hashing

6.3.4 roi

The `ROIManager` class is used to manage and save/load ROI positions.

```
In [1]: from drymass.roi import ROIManager

In [2]: rmg = ROIManager(identifier="example")

In [3]: rmg.add(roi_slice=(slice(10, 100), slice(50, 140)),
   ...:         image_index=2,
   ...:         roi_index=0,
   ...:         identifier="example_subroi_2.0")
   ...:

In [4]: rmg.get_from_image_index(2)
Out[4]: [example_subroi_2.0        2        0        (slice(10, 100, None), slice(50,␣
↪140, None))]
```

Saving and loading can be done with:

```
In [5]: rmg.save("output_rois.txt")

In [6]: rmg2 = ROIManager(identifier="ex")

In [7]: rmg2.load("output_rois.txt")
```

And the content of "output_rois.txt" is:

```
example_subroi_2.0      2       0       (slice(10, 100, None), slice(50, 140,␣
↪None))
```

exception drymass.roi.**ROIManagerWarning**
Used for unexpected keyword arguments.

CHAPTER 7

Changelog

List of changes in-between DryMass releases.

7.1 version 0.10.2

- enh: display progress in percent when using the CLI

7.2 version 0.10.1

- setup: bump qpfromat from 0.10.3 to 0.10.4
- setup: bump qpimage from 0.6.0 to 0.6.1
- setup: bump qpsphere frmo 0.5.5 to 0.5.7

7.3 version 0.10.0

- ref: migrate to scikit-image 0.16.1

7.4 version 0.9.4

- fix: correctly sort ignored ROIs in config (previously used float-representation, now use version-representation)

7.5 version 0.9.3

- fix: correctly sort ignored ROIs in config
- enh: reuse fits from previous runs in "dm_analyze_sphere"
- enh: print number of reused fits in "dm_analyze_sphere"
- ref: changed convention for QPSeries and QPImage data identifiers (QPSeries contains full trace of hashes, QPImage only contains dataset hash, image index, and simulation type)
- docs: update tutorials (rerender figures and link to figshare data)

7.6 version 0.9.2

- fix: fitted phase images with the Rytov approximation sometimes exhibited 2PI phase offsets (bump qpimage from 0.5.4 to 0.6.0)

7.7 version 0.9.1

- fix: skip sphere analysis when bad model parameters are detected
- fix: allow to force-regenerate the sphere analysis images file by deleting it prior to running "dm_analyze_sphere"
- fix: do not plot profile in sphere analysis when the center position not within the image data area
- setup: bump qpsphere from 0.5.3 to 0.5.4
- tests: fix failed test due to previous refactorization

7.8 version 0.9.0

- feat: allow to change the thresholding filter or use a manual threshold
- feat: new thresholding filter "dm-nuclei" designed for HL60 cell nuclei segmentation
- fix: configuration file was "touched" even though it was only read
- setup: bump qpsphere from 0.5.2 to 0.5.3
- ref: increase verbosity of error messages when edge detection step fails
- ref: new *threshold* submodule for all thresholding operations

7.9 version 0.8.8

- setup: bump qpformat from 0.10.2 to 0.10.3 (changes dataset identifiers)
- setup: bump qpsphere from 0.5.1 to 0.5.2

7.10 version 0.8.7

- setup: pin numpy<1.16.0 because of pinned scikit-image==0.14.0

7.11 version 0.8.6

- docs: minor improvements
- setup: pin scikit-image version again to 0.14.0 until the threshold_li issue is resolved once and for all

7.12 version 0.8.5

- docs: minor update

7.13 version 0.8.4

- enh: bump dependencies on qpformat, qpimage, and qpsphere
  - improved file format support
  - improved speed of in-memory operations
  - several other minor fixes
- docs: minor cleanup

7.14 version 0.8.3

- enh: when using incomplete profile files, they are now merged with (instead of overriding) existing profiles (drymass.cfg) in the respective output directories (#27)
- docs: minor update

7.15 version 0.8.2

- fix: pin scikit-image version to 0.14.0 because threshold_li has changed (https://github.com/scikit-image/scikit-image/issues/3605)
- enh: implement "roi" configuration option "force" (#8)
- ref: replace default config values of *nan* with *None* (#13)
- ref: turn comments in definitions.py into real strings (#23)
- docs: add automation tutorial (usage of profiles and scripts)

7.16 version 0.8.1

- fix: replaced hack for plotting units with better solution due to incompatibility with tight_layout in matplotilb 3.0.2

7.17 version 0.8.0

- feat: implement profile management via *dm_profile* command and allow to select profiles via the "–profile" command-line argument in the command-line interface (#20)
- fix: if time is unknown, write "nan" instead of "0" to the output statistics file
- ref: explicitly use row-column coordinates in skimage's regionprops (bump skimage version to 0.14.1)

7.18 version 0.7.3

- fix: do not create "_dm" results directory for root directory in recursive analysis mode (#19)
- docs:
  - add DryMass logo
  - update software description

7.19 version 0.7.2

- skipped

7.20 version 0.7.1

- maintenance release

7.21 version 0.7.0

- feat: support recursive analysis with the "–recursive" command-line argument (#15)

7.22 version 0.6.2

- fix: regression in 0.6.0 ImageJ cannot open compressed hyperstack data; disabling compression for now (see #14)

7.23 version 0.6.1

- fix: MetaDataMissingError for some data types (qpformat 0.4.3)

7.24 version 0.6.0

- fix: error when removing ROIs that overlap with two ROIs in the background image
- feat: allow to exclude individual ROIs in drymass.cfg (#5)
- enh: improve error message verbosity
- enh: use compression for TIFF output
- enh: use new class for handling ROIs internally (#9)
- enh: start counting of ROIs and sensor images at 1 instead of 0 (#11)
- ref: handle 'None' values directly in config.ConfigFile

7.25 version 0.5.0

- feat: automatically extract meta data from QPImage/QPSeries data (qpformat 0.4.0)
- fix: setting "dist border px" to 0 still removed objects from border
- fix: scikit-image methods could not be used for "binary threshold"
- fix: set "image fix phase offset" to True by default, otherwise objects in the background cause a false offset
- improvement: optimize CLI plots
- docs:
  - update tutorials with new CLI plots
  - add tutorial for background correction (#3)

7.26 version 0.4.1:

- fix: extracted wrong phase image from SID4Bio data (qpformat 0.3.5)

7.27 version 0.4.0

- feat: ROI background correction supports masking of the imaged object via the "amplitude mask sphere" and "phase mask sphere" settings in the [bg] section.
- setup: support qpimage>=0.4.0
- tests: improve coverage
- docs: minor update (#7)

7.28 version 0.3.3

- ci: automate PyPI release with travis-ci
- setup: bump versions of dependent packages

7.29 version 0.3.2

- docs: update of introduction and minor changes

7.30 version 0.3.1

- bump version because of regression fixed in qpformat 0.2.1
- feat: flush statistics file during writing

7.31 version 0.3.0

- drop support for Python 3.5
- ref: background correction in convert.py uses background indices starting at 0 while the CLI keeps using indices starting at 1
- fix: hologram keyword arguments were not used for background correction
- fix: case where index-based background correction does not work
- feat: sphere analysis is picked up where left off in previous run
- command line interface:
  - fix: allow white spaces for input data path argument
  - fix: improve verbosity of status messages
- docs: add troubleshooting section

7.32 version 0.2.0

- feat: reuse computed ROI data from previous runs
- ref: update [roi] configuration key names to reflect pixel units: "dist border px", "exclude overlap px", "pad border px"
- fix: improve verbosity of error messages
- fix: config value "[roi] size variation" must be in interval [0,1]
- fix: handle ROIs with failed edge detection in "dm_analyze_sphere"

7.33 version 0.1.4

- feat: add local thresholding step before segmentation
- fix: check number of qpimages in sensor_data.h5 during dm_convert
- fix: "exclude overlap" only used for background image
- fix: not possible to analyze phasics data folder when raw data is present (qpformat 0.1.5)

7.34 version 0.1.3

- automatically fix inverted ranges when loading "roi_slices.txt"
- remove "holo" section from drymass.cfg if not relevant
- allow user-defined ROI selection via "[roi] enabled = False"
- added hologram analysis parameter tuning (qpimage 0.1.6)
- added 2D model fitting to sphere analysis (qpsphere 0.1.4)
- fixed *cfg_funcs.int_or_str*

7.35 version 0.1.2

- allow to disable sensor image tif export
- support input QPI data with different shapes
- fixed wrong assumption about definition of dry mass in cells (water vs. intracellular salt solution)
- allow to use separate file or series index for background correction with experimental data (#6)
- small visualization improvements

7.36 version 0.1.1

- add file name to image identifier and update plotting (#4)
- renamed "object" to "identifier" in statistics output
- binary-based background correction (manual or automated threshold)
- allow to disable background correction with "[bg] enabled = False"
- ask for missing meta data keys in CLI only when they are required

7.37 version 0.1.0

- initial release

CHAPTER 8

Bilbliography

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Bibliography

[Bar52] R. Barer. Interference Microscopy and Mass Determination. *Nature*, 169(4296):366–367, 1952. doi:10.1038/169366b0.

[Bar53] R. Barer. Determination of dry mass, thickness, solid and water concentration in living cells. *Nature*, 172:1097–1098, 1953. doi:10.1038/1721097a0.

[Bar57] R. Barer. Refractometry and interferometry of living cells. *Journal of the Optical Society of America*, 47(6):545, jun 1957. doi:10.1364/josa.47.000545.

[BJ54] R. Barer and S. Joseph. Refractometry of Living Cells Part I. Basic Principles. *Quarterly Journal of Microscopical Science*, 95(4):399–423, dec 1954. doi:10.1038/171720a0.

[HGC94] Y. Harpaz, M. Gerstein, and C. Chothia. Volume changes on protein folding. *Structure*, 2(7):641–649, jul 1994. doi:10.1016/s0969-2126(00)00065-4.

[KKL+07] B. Kemper, S. Kosmeier, P. Langehanenberg, G. von Bally, I. Bredebusch, W. Domschke, and J. Schnekenburger. Integral refractive index determination of living suspension cells by multifocus digital holographic phase contrast microscopy. *Journal of Biomedical Optics*, 12(5):054009, 2007. doi:10.1117/1.2798639.

[KvB07] B. Kemper and G. von Bally. Digital holographic microscopy for live cell applications and technical inspection. *Applied Optics*, 47(4):A52, oct 2007. doi:10.1364/ao.47.000a52.

[LL02] M. Lehmann and H. Lichte. Tutorial on off-axis electron holography. *Microscopy and Microanalysis*, 8(6):447–466, dec 2002. doi:10.1017/s1431927602020147.

[MSG+18] P. Müller, M. Schürmann, S. Girardo, G. Cojoc, and Guck J. Accurate evaluation of size and refractive index for spherical objects in quantitative phase imaging. *Optics Express*, 26(8):10729–10743, 2018. doi:10.1364/OE.26.010729.

[MSGG19] Paul Müller, Mirjam Schürmann, Salvatore Girardo, and Jochen Guck. Reference quantitative phase microscopy images of spherical objects. *figshare*, Sep 2019. doi:10.6084/m9.figshare.9879326.v1.

[SCG+17] M. Schürmann, G. Cojoc, S. Girardo, E. Ulbricht, J. Guck, and P. Müller. Three-dimensional correlative single-cell imaging utilizing fluorescence and refractive index tomography. *Journal of Biophotonics*, 11(3):e201700145, aug 2017. doi:10.1002/jbio.201700145.

[SSM+15] M. Schürmann, J. Scholze, P. Müller, C. J. Chan, A. E. Ekpenyong, K. J. Chalut, and J. Guck. Chapter 9 - Refractive index measurements of single, spherical cells using digital holographic microscopy. In Ewa K Paluch, editor, *Biophysical Methods in Cell Biology*, volume 125 of Methods in Cell Biology, pages 143–159. Academic Press, 2015. doi:10.1016/bs.mcb.2014.10.016.

[SSM+16] M. Schürmann, J. Scholze, P. Müller, J. Guck, and C. J. Chan. Cell nuclei have lower refractive index and mass density than cytoplasm. *Journal of Biophotonics*, 9(10):1068–1076, oct 2016. doi:10.1002/jbio.201500273.

Python Module Index

d

Index

L

M

O

P

R

S

T

U