ImageViewer

Release 1

Melissa Lajtos

Feb 29, 2020

CONTENTS:

CHAPTER

ONE

IMAGEVIEWER

1.1 imageviewer package

1.1.1 Subpackages

imageviewer.tests package

Submodules

imageviewer.tests.test module

class imageviewer.tests.test.**TestAddData**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Class for testing add_data() method of the ImageViewer class, which simply calls the add_data() method of the DataHandling class.

I am not sure if these kinds of tests make much sense, since in the tested functions only numpy functions are called and nothing should go wrong by that.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

setUp()
Hook method for setting up the test fixture before exercising it.

test_different_real_and_imag_2x2_float()
Testing the method DataHandling.add_data() called by ImageViewer.add_data() with a one slice of data containing complex numbers x + iy where x and y are floats. :return: None.

test_different_real_and_imag_3_2x2_float()
Testing the method DataHandling.add_data() called by ImageViewer.add_data() with a one slice of data containing complex numbers x + iy where x and y are floats. :return: None.

test_same_real_and_imag_2x2_float()
Testing the method DataHandling.add_data() called by ImageViewer.add_data() with a one slice of data containing complex numbers x + ix where x is a float. :return: None.

test_same_real_and_imag_2x2_int()
Testing the method DataHandling.add_data() called by ImageViewer.add_data() with a one slice of data containing complex numbers x + ix where x is an integer (case where x=0 occurs). :return: None.

test_same_real_and_imag_2x3_float()
Testing the method DataHandling.add_data() called by ImageViewer.add_data() with a one slice of data containing complex numbers x + ix where x is a float. :return: None.

class imageviewer.tests.test.**TestImageViewer**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Class for testing basic settings and behaviour of the ImageViewer class.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

setUp()
Hook method for setting up the test fixture before exercising it.

test_colormap_change()
Test setting different colormap by triggering action in menuColormap. Only one action should be checked at a time. :return:

test_defaults()
Test default values that should be set when creating an instance of ImageViewer. :return: None.

imageviewer.tests.test.**create_custom_complex_2dim_data**(*real*, *imaginary*)

imageviewer.tests.test.**create_custom_complex_3dim_data**(*real*, *imaginary*, *n*)

Module contents

1.1.2 Submodules

1.1.3 imageviewer.fileHandling module

class imageviewer.fileHandling.**GetFileContent**(*selected*)
Bases: PyQt5.QtCore.QRunnable

This class serves as a parent class for other classes which will handle different file types. It inherits from QRunnable, thus it will be called in a separate thread.

Parameters **selected** (*str*) – The name of the selected dataset data shall be loaded from.

class imageviewer.fileHandling.**GetFileContentDicom**(*file_sets*, *selected*, *directory*)
Bases: *imageviewer.fileHandling.GetFileContent*

Class for loading dicom data. Inherits from *GetFileContent*.

Parameters

- **file_sets** (*list[list[str]]*) – Filesets identified by *IdentifyDatasetsDicom*.
- **selected** (*str*) – The name of the first file of the selected dataset within the directory.
- **directory** (*str*) – The selected directory.

run()
Loads the selected dataset into an array and passes it to self.signals.add_data.emit(). Emits finished signal after that.

Gets called when the thread is started.

class imageviewer.fileHandling.**GetFileContentH5**(*filename*, *selected*)
Bases: *imageviewer.fileHandling.GetFileContent*

Class for loading h5 data. Inherits from *GetFileContent*.

Parameters

- **filename** (*str*) – The name of the selected file.
- **selected** (*str*) – The name of the selected dataset within the file. If the file only contains one dataset, this needs to be the same as parameter `filename`.

run()
Loads the selected dataset into an array and passes it to self.signals.add_data.emit(). Emits finished signal after that.

Gets called when the thread is started.

class imageviewer.fileHandling.**GetFileContentSignals**
Bases: `PyQt5.QtCore.QObject`

Class for generating thread signals for *GetFileContent*.

add_data

finished

class imageviewer.fileHandling.**IdentifyDatasetsDicom**(*filenames*)
Bases: `PyQt5.QtCore.QRunnable`

Class for identifying files belonging together (forming a dataset) within a dicom folder. It inherits from QRunnable, thus it will be called in a separate thread.

Signals are from the *IdentifyDatasetsDicomSignals* class.

Parameters **filenames** (*list[str]*) – The names of all the files within the dicom directory.

run()
This function looks at all filenames in parameter filenames separately and sorts them into sets.

It does so by comparing a filename to the next filename: if the filenames differ at at least two characters and the first and last different characters are more than 2 indices apart, they are considered to be of different sets. This works on the assumption that the filenames contain ongoing numbers, one representing dataset and one representing file (slice) within dataset. Using 2 indices as the criteria, this approach may be a problem when there are more than 1000 files within a set.

Gets called when the thread is started.

class imageviewer.fileHandling.**IdentifyDatasetsDicomSignals**
Bases: `PyQt5.QtCore.QObject`

Class for generating thread signals for the *IdentifyDatasetsDicom* class.

setsIdentified

1.1.4 imageviewer.main module

class imageviewer.main.**DataHandling**
Bases: object

Image data is stored in this class sorted by magnitude and phase.

Variables

- **data** (*numpy.ndarray*) – Contains the original image data from the file (squeezed if there was an unnecessary dimension).
- **magn_values** (*numpy.ndarray*) – The magnitude values of the image data if only one slice is present, else it is 0.
- **phase_values** (*numpy.ndarray*) – The phase values of the image data if only one slice is present, else it is 0.
- **magn_slices** (*numpy.ndarray*) – The magnitude values of the image data if multiple slices are present, else it is 0.
- **phase_slices** (*numpy.ndarray*) – The phase values of the image data if multiple slices are present, else it is 0.

add_data(*data*)
This function takes the data from a loaded file, processes it, and stores it in the right instance attributes.

The data gets squeezed in order to remove any unnecessary dimension. After that the number of dimensions gets checked. If 2, the data contains only one slice and gets stored in `magn_values` and `phase_values`, if 3, the data contains multiple slices and gets stored in `magn_slices` and `phase_slices`.

Parameters **data** (*numpy.ndarray*) – Image data loaded from file.

clear_data()
Sets all attributes back to 0 as they were after initialization.

show_data()
Prints type and shape of `data`.

class imageviewer.main.**ImageViewer**
Bases: PyQt5.QtWidgets.QMainWindow, imageviewer.ui.mainWindow.Ui_MainWindow

Main class for showing the UI. Runs in the main thread.

Lets user open h5 and dicom files of which image data will be displayed using matplotlib, change the colormap, and select if magnitude or phase shall be displayed.

Variables

- **filename** (*instance of h5py._hl.files.File (in case of h5), or list[str] (in case of dicom)*) – Either the h5 file the user selected, or a list of the names of the first files of all dicom dataset within a dicom directory.
- **directory** (*str*) – Whole path of the dicom directory the user selected.
- **dicom_sets** (*list[list[str]]*) – List of lists with all files belonging to the same dataset within the dicom directory, identified by *IdentifyDatasetsDicom*.
- **slice** (*int*) – The number of the slice of the image data being displayed.
- **cmap** (*str*) – Name of the colormap (matplotlib) used to plot the data.
- **select_box** (*SelectBox*) – Window which lets user select a dataset within a selected file/directory.

- **data_handling** (*DataHandling*) – Data is being processed and stored here.
- ***mplwidget*** (MplWidget) – A selfmade widget (inherits from QWidget) which is used to visualize image data.

add_data(*data*)
Hands the data over to *DataHandling* to store it appropriately.

Parameters **data** (*numpy.ndarray*) – Image data from file.

browse_folder_dcm()
Opens a file dialog for selecting a dicom folder. Once a folder is selected, it is stored in directory. If there is more than one file present within this directory (which is usually the case), a new thread, started by *IdentifyDatasetsDicom*, will sort the files into datasets. Once sorting is done, it will call *open_file_dicom()*.

browse_folder_h5()
Opens a file dialog for selecting an .h5 file. Once a file is selected, it is stored in filename and *open_file_h5()* gets called.

change_cmap()
Is called when user changes the colormap. Sets cmap to selected colormap and calls *plot_data_if_data()*.

close()
Exits the application.

done()
As of now, this only calls *plot_data()*. Maybe in the future this will fulfill a certain purpose, therefore it is kept for now.

open_file_dicom(*file_sets*)
Checks if there is more than one dataset within the file_sets and opens instance of *SelectBox* which lets user select a dataset.

It sets dicom_sets to file_sets and filename to a list of the names of the first files of each fileset. These names will then stand for the set the file belongs to respectively.

Parameters **file_sets** (*list[list[str]]*) – A list that contains a list with the names of all files of a fileset for each fileset.

open_file_h5()
Checks if there is more than one dataset within the file (filename) to open; if yes, opens instance of *SelectBox* which lets user select a dataset and will call *read_data()*, if no, creates instance of *GetFileContentH5* which will run in a new thread to get the data within the file.

plot_data()
Responsible for plotting the image data correctly.

This function checks if there is data stored as slices or one slice only and plots the data accordingly on the mplwidget. It also sets slice_label's text.

plot_data_if_data()
This method calls *plot_data()* if any image data is given in data_handling.data, else it does nothing.

read_data()
Checks if the file selected via *SelectBox* is of type h5 or dicom and starts the matching thread (*GetFileContentH5* or *GetFileContentDicom*) to load the data.

wheelEvent(*event*)
This function enables going through the data slices (if there are ones) using the mouse wheel. It turns a

120° turn in the y direction of the mousewheel into one slice difference. This function only does something if there are data slices given.

Parameters **event** (*QWheelEvent*) – The wheel event.

class imageviewer.main.**SelectBox**

Bases: PyQt5.QtWidgets.QMainWindow, imageviewer.ui.selectBox.Ui_MainWindow

Window for selecting the desired dataset within an h5 file or dicom folder.

Variables **selected** (*str*) – Name of the selected file within the UI window.

cancel()

Closes the window and sets selected back to *None*.

confirm()

Stores the name of the selected dataset in SelectBox.selected and closes the window.

imageviewer.main.**main**()

1.1.5 Module contents

class imageviewer.ui.mplwidget.**MplWidget**(*parent=None*)

Selfmade widget used to visualize image data.

Variables **canvas** (matplotlib.backends.backend_qt5agg.FigureCanvasQTAgg) – The actual matplotlib figure canvas where data is plotted.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

i