Image Filters, Projections and Slices

Generated by Doxygen 1.9.6

Chapter 1

How to use the software

We have 3 executable files (We only have students with macbooks. That is why we can only upload the mac executable):

- `./main` → The main programm to apply filters, projections, ...

```
g++-12 src/main.cpp src/UserInterface.cpp src/Image.cpp src/Filter.cpp src/Volume.cpp src/Slice.cpp
src/Util.cpp src/Projection.cpp -o main
```

- `./test` → Tests the functions of the main programm.

```
g++-12 src/test.cpp src/Image.cpp src/Filter.cpp src/Volume.cpp src/Slice.cpp src/Testing.cpp
src/Util.cpp src/Projection.cpp -o test
```

- `./time` → Times the functions of the main programm.

```
g++-12 src/time.cpp src/Image.cpp src/Filter.cpp src/Volume.cpp src/Slice.cpp src/Util.cpp
src/Projection.cpp -o time
```

1.1 Main

When the users starts the programme, the user is asked for input data in the console. There are two possibilities. On the one hand, a path to an image can be specified. Then the data type is recognised as a 2D image. If a path to a folder is specified, it is assumed that this is a volume. The path should be inside the root directory so that the programm has access to the file/directory.

```
Please input the path to the input data.
- For 2D images specify a path to an image (for example 'Images/gracehopper.png').
- For 3D volumes specify a path to a folder (for example 'Scans/confuciusornis').

>
```

1.1.1 2D Image

If a path to an image is specified, a selection of all possible filters is displayed. If the user wants to select a filter, the user must enter the number in front of the respective filter.

```
Please input the number of the 2D filter you want to apply.
[0] Apply selected filter

[1] greyscale
[2] brightness
[3] median-blur
[4] gaussian-blur
[5] sobel
[6] prewitt

Selected filters: []
> 1
```

After pressing enter the selcted Filter will be displayed under "Selected filters".

```
Please input the number of the 2D filter you want to apply.
[0] Apply selected filter

[1] greyscale
[2] brightness
[3] median-blur
[4] gaussian-blur
[5] sobel
[6] prewitt

Selected filters: [greyscale]
>
```

If the user wants to add another Filter, that will be applied after the first filter, he can enter another number. He can specify as many filters as he wants.

```
Please input the number of the 2D filter you want to apply.
[0] Apply selected filter

[1] greyscale
[2] brightness
[3] median-blur
[4] gaussian-blur
[5] sobel
[6] prewitt

Selected filters: [greyscale -> prewitt]
>
```

After the user has selected all filters that he wants to apply on the original image he has to press "0". Now the first filter will be applied to the originial image. Then the second filter is applied to the output of the previous filter and so on. If the filter requires additional input from the user like the kernel size, another query will appear on the screen that will ask the user for more input. The final Image will be stored in the output directory. The programm will create a new directory inside this directory if it does not already exists. The name of this directory is the name of the original image. Inside of this folder, the final image will be stored. The name of this image represents all the filters that were applied to the image.

1.1.2 Volume

If a volume is specified, the user is first asked whether all images in the specified folder are to be used.

```
Do you want to use all the images in that folder?

[1] Yes
[2] No

>
```

If the user selects "No", only a thin slab of the whole volume is used. To save time, the remaining part is not read in at all into memory. To define the range, the z coordinate must be specified at which the volume should start to be

read in. For example, a z-coordinate of 10 corresponds to the tenth image.

```
Please input the first z-value:
10
```

Then the second coordinate must be entered up to which the volume is to be read in.

```
Please input the second z-value:
100
```

After the volume has been read in, various options are suggested which can be applied to the volume.

```
Please input a number between 1 and 3!:

[1] Slice
[2] Projection
[3] 3D-Filter & Projection

>
```

1.1.2.1 Slicing

If "Slicing" is selected, the user can slice the volume in a different plane. There are two choices. By typing the number in front of the options, they can be selected.

```
Please input the number of the plane:

[1] x-z plane
[2] y-z plane

>
```

After that the user must specify the constant value. If the user selected the "x-z plane" option, he needs to input a y value. If the user selected the "y-z plane" he has to input a x value.

```
Please input the constant y-value:
>
```

The output will be saved like with 2D-Images. However the filename of the resulting image is different. It will be for example: "x-z plane_20.png".

1.1.2.2 Projection

When the user selects "Projection", there are 4 different projections to choose from. By entering the number in front of the respective projection and pressing enter, it can be selected.

```
Please input the Projection you want to apply:

[1] Maximum intensity projection (MIP)
[2] Minimum intensity projection (MinIP)
[3] Mean - Average intensity projection (AIP)
[4] Median - Average intensity projection (AIP)

>
```

The output will be saved similar to the 2D-Images or Slicing but with a different name. For example: "Maximum intensity projection (MIP).png"

1.1.2.3 3D-Filter & Projection

When the user selects "3D-Filter & Projection", the system first asks for the 3D-filter which should be applied. The filter can be selected by typing in the number in front of the corresponding option.

```
Please input the 3D Filter you want to apply:

[1] 3D Gaussian
[2] 3D Median

>
```

Then the user will be asked for the projection which should be applied (see previous point). After selecting the projection, the user will also be asked for the kernel size for the selected 3d filter. The output will be saved similar to the 2D-Images, Slicing or Projection but with a different name.

1.2 Test

By executing `./test` you get an overview of all tests whether they were successful or not. They are also sorted thematically. The tests do not take much time to run because only small amounts of data are used.

1.3 Timing

By executing `./time`, you get the average time that the programme/function needs when repeated several times. Since the execution can take a little longer, we have listed the output here, which was obtained on one of our laptops.

1.3.0.1 Timings of 2D-Filters:

- Timings for image 'Images/gracehopper.png':
  Avg. timing of greyscale: **5ms**
  Avg. timing of brightness: **5ms**
  Avg. timing of medianblur (5x5): **1142ms**
  Avg. timing of medianblur (7x7): **2583ms**
  Avg. timing of gaussianblur (5x5): **268ms**
  Avg. timing of gaussianblur (7x7): **588ms**
  Avg. timing of sobel: **15ms**
  Avg. timing of prewitt: **104ms**

- Timings for image 'Images/tienshan.png':
  Avg. timing of greyscale: **55ms**
  Avg. timing of brightness: **60ms**
  Avg. timing of medianblur (5x5): **12030ms**
  Avg. timing of medianblur (7x7): **27280ms**
  Avg. timing of gaussianblur (5x5): **2737ms**
  Avg. timing of gaussianblur (7x7): **6001ms**
  Avg. timing of sobel: **162ms**
  Avg. timing of prewitt: **1051ms**

1.3.0.2 Timings of Slicing:

- Timings for volume 'Scans/confuciusornis with num images: 20:
  Avg. timing of XZ Slicing: **0ms**
  Avg. timing of YZ Slicing: **0ms**

- Timings for volume 'Scans/fracture with num images: 20:
  Avg. timing of XZ Slicing: **0ms**
  Avg. timing of YZ Slicing: **0ms**

- Timings for volume 'Scans/confuciusornis with num images: 200:
  Avg. timing of XZ Slicing: **1ms**
  Avg. timing of YZ Slicing: **15ms**

- Timings for volume 'Scans/fracture with num images: 200:
  Avg. timing of XZ Slicing: **1ms**
  Avg. timing of YZ Slicing: **2ms**

1.3.0.3 Timings of 3D-Projections:

- Timings for volume 'Scans/confuciusornis with num images: 20:
  Avg. timing of maximum intensity projection: **239ms**
  Avg. timing of minimum intensity projection: **224ms**
  Avg. timing of mean intensity projection: **177ms**
  Avg. timing of median intensity projection: **2208ms**

- Timings for volume 'Scans/fracture with num images: 20:
  Avg. timing of maximum intensity projection: **44ms**
  Avg. timing of minimum intensity projection: **44ms**
  Avg. timing of mean intensity projection: **29ms**
  Avg. timing of median intensity projection: **392ms**

- Timings for volume 'Scans/confuciusornis with num images: 200:
  Avg. timing of maximum intensity projection: **6839ms**
  Avg. timing of minimum intensity projection: **6666ms**
  Avg. timing of mean intensity projection: **6957ms**
  Avg. timing of median intensity projection: **26237ms**

- Timings for volume 'Scans/fracture with num images: 200:
  Avg. timing of maximum intensity projection: **466ms**
  Avg. timing of minimum intensity projection: **485m**
  Avg. timing of mean intensity projection: **435ms**
  Avg. timing of median intensity projection: **3052ms**

1.3.0.4 Timings of 3D-Filters:

- Timings for volume 'Scans/confuciusornis with num images: 20:
  Avg. timing (projected) of the median 3d filter (5x5): **266s**
  Avg. timing (projected) of the gaussian 3d filter (5x5): **38s**

- Timings for volume 'Scans/fracture with num images: 20:
  Avg. timing (projected) of the median 3d filter (5x5): **36s**
  Avg. timing (projected) of the gaussian 3d filter (5x5): **6s**

- Timings for volume 'Scans/confuciusornis with num images: 200:
  Avg. timing (projected) of the median 3d filter (5x5): **2660s**
  Avg. timing (projected) of the gaussian 3d filter (5x5): **380s**

- Timings for volume 'Scans/fracture with num images: 200:
  Avg. timing (projected) of the median 3d filter (5x5): **360s**
  Avg. timing (projected) of the gaussian 3d filter (5x5): **60s**

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Filter Class Reference

Class with static methods to apply filters. The Class contains 2D and 3D Filters.
`#include <Filter.h>`

Static Public Member Functions

- static void applyGreyscaleFilter (Image &image, int input=-1)

  Applies greyscale filter on an image, change the image color to grey.
- static void Brightness (Image &image, int value)

  Applies the Brightness algorithm on an image to get brighter or darker one.
- static void applyMedianBlurFilter (Image &image, int input=-1)

  Applies the median blur filter algorithm on an image.
- static void applyGaussianBlurFilter (Image &image, int kernel_size=-1)

  Applies the gaussian blur filter algorithm on an image.
- static void sobel (Image &image, int input=-1)

  Applies the sobel filter.
- static void prewitt (Image &image, int input=-1)

  Applies the prewitt edge detection algorithm on an image.
- static void **gaussian3D** (Volume &volume, int input=-1)
- static void median3D (Volume &volume, int input=-1)

  Applies the 3D median blur filter algorithm on an image.

5.1.1 Detailed Description

Class with static methods to apply filters. The Class contains 2D and 3D Filters.

5.1.2 Member Function Documentation

5.1.2.1 applyGaussianBlurFilter()

```
void Filter::applyGaussianBlurFilter (
            Image & image,
            int kernel_size = -1 )  [static]
```

Applies the gaussian blur filter algorithm on an image.

Parameters

image	the reference of the image that should be used for the filter.
input	kernel size, eg: 5x5(input=5) or 7x7(input=7)

5.1.2.2 applyGreyscaleFilter()

```
void Filter::applyGreyscaleFilter (
            Image & image,
            int input = -1 )  [static]
```

Applies greyscale filter on an image, change the image color to grey.

Parameters

image	the reference of the image that should be used for the filter.
input	Because the greyscale filter does not has an additional input, it should be always -1.

5.1.2.3 applyMedianBlurFilter()

```
void Filter::applyMedianBlurFilter (
            Image & image,
            int input = -1 )  [static]
```

Applies the median blur filter algorithm on an image.

Parameters

image	the reference of the image that should be used for the filter.
input	kernel size, eg:3x3(input=3), 5x5(input=5), 7x7(input=7), etc.

5.1.2.4 Brightness()

```
void Filter::Brightness (
            Image & image,
            int value )  [static]
```

Applies the Brightness algorithm on an image to get brighter or darker one.

Parameters

image	the reference of the image that should be used for the filter.
value	the user give the value that will be added to all pixels, positive value for a brighter image and negative value for a darker image

5.1.2.5 median3D()

```
void Filter::median3D (
            Volume & volume,
            int input = -1 )  [static]
```

Applies the 3D median blur filter algorithm on an image.

Parameters

volume	the reference of the volume that should be used for the filter.
input	kernel size, eg:3x3x3(input=3), 5x5x5(input=5), 7x7x7(input=7), etc.

5.1.2.6 prewitt()

```
void Filter::prewitt (
            Image & image,
            int input = -1 )  [static]
```

Applies the prewitt edge detection algorithm on an image.

Parameters

image	the reference of the image that should be used for the filter.
input	Because the prewitt filter does not has an additional input, it should be always -1.

5.1.2.7 sobel()

```
void Filter::sobel (
            Image & image,
            int input = -1 )  [static]
```

Applies the sobel filter.

Parameters

image	the reference of the image that should be used for the filter.
input	parameter used to comply with user interface. Does not affect the algorithm.

The documentation for this class was generated from the following files:

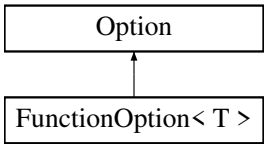
- src/Filter.h
- src/Filter.cpp

5.2 FunctionOption< T > Struct Template Reference

A struct that represents an option that a User can select in the UserInterface. When the user selects this option the specified function will be called.

`#include <UserInterface.h>`

Inheritance diagram for FunctionOption< T >:



Public Attributes

- std::function< void(T &, int)> function
- std::string additionalString = ""
- int input

Public Attributes inherited from Option

- std::string name

5.2.1 Detailed Description

template<class T>
struct FunctionOption< T >

A struct that represents an option that a User can select in the UserInterface. When the user selects this option the specified function will be called.

Template Parameters

T	The first input type to the function.

5.2.2 Member Data Documentation

5.2.2.1 additionalString

```
template<class T >
std::string FunctionOption< T >::additionalString = ""
```

If a String is specified, the string will be displayed to the user and will ask him for an input (integer)

5.2.2.2 function

```
template<class T >
std::function<void(T&, int)> FunctionOption< T >::function
```

The function that sould be called When the user selects this Options. The function has two parameters. One arbitrary data type and an integer.

5.2.2.3 input

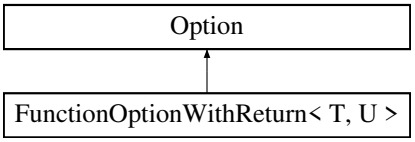
```
template<class T >
int FunctionOption< T >::input
```

Saves the input of the integer.
The documentation for this struct was generated from the following file:

- src/UserInterface.h

5.3 FunctionOptionWithReturn< T, U > Struct Template Reference

A struct that represents an option that a User can select in the UserInterface. When the user selects this option the specified function will be called. The function can return something. The function has two parameters. One arbitrary data type and an integer.
`#include <UserInterface.h>`
Inheritance diagram for FunctionOptionWithReturn< T, U >:



Public Attributes

- std::function< U(T &, int)> function
- std::string additionalString = ""
- int input

Public Attributes inherited from Option

- std::string name

5.3.1 Detailed Description

template<class T, class U>
struct FunctionOptionWithReturn< T, U >

A struct that represents an option that a User can select in the UserInterface. When the user selects this option the specified function will be called. The function can return something. The function has two parameters. One arbitrary data type and an integer.

Template Parameters

T	the first input type to the function.
U	the return type of the function.

5.3.2 Member Data Documentation

5.3.2.1 additionalString

```
template<class T , class U >
std::string FunctionOptionWithReturn< T, U >::additionalString = ""
```

If a String is specified, the string will be displayed to the user and will ask him for an input (integer)

5.3.2.2 function

```
template<class T , class U >
std::function<U(T&, int)> FunctionOptionWithReturn< T, U >::function
```

The function that should be called when the user selects this Options.

5.3.2.3 input

```
template<class T , class U >
int FunctionOptionWithReturn< T, U >::input
```

Saves the input of the integer.
The documentation for this struct was generated from the following file:

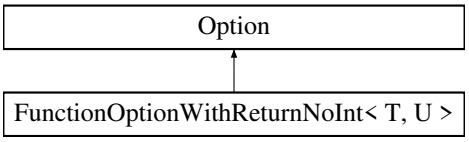
- src/UserInterface.h

5.4 FunctionOptionWithReturnNoInt< T, U > Struct Template Reference

A struct that represents an option that a User can select in the UserInterface. When the user selects this option the specified function will be called. The function can return something. The function has one parameter with an arbitrary data type.

```
#include <UserInterface.h>
```

Inheritance diagram for FunctionOptionWithReturnNoInt< T, U >:



Public Attributes

- std::function< U(T &)> function
- std::string additionalString = ""
- int input

Public Attributes inherited from Option

- std::string name

5.4.1 Detailed Description

template<class T, class U>
struct FunctionOptionWithReturnNoInt< T, U >

A struct that represents an option that a User can select in the UserInterface. When the user selects this option the specified function will be called. The function can return something. The function has one parameter with an arbitrary data type.

Template Parameters

T	the first input type to the function.
U	the return type of the function.

5.4.2 Member Data Documentation

5.4.2.1 additionalString

```
template<class T , class U >
std::string FunctionOptionWithReturnNoInt< T, U >::additionalString = ""
```

If a String is specified, the string will be displayed to the user and will ask him for an input (integer)

5.4.2.2 function

```
template<class T , class U >
std::function<U(T&)> FunctionOptionWithReturnNoInt< T, U >::function
```

The function that sould be called when the user selects this Options.

5.4.2.3 input

```
template<class T , class U >
int FunctionOptionWithReturnNoInt< T, U >::input
```

Saves the input of the integer.
The documentation for this struct was generated from the following file:

- src/UserInterface.h

5.5 Image Class Reference

The class saves the data of one image. It has different getters and setters that can be used to access or set the data of the image.
`#include <Image.h>`

Public Member Functions

- Image (int width, int height, int channels)

  Creates an image with a defined width, height and number of channels. It does not preallocate the memory.
- Image (const char ∗filename)

  It loads one image and saves the data in an 1D-Array.
- const unsigned char ∗ constDataGetter () const

  Returns the data. The data can not be changed.
- unsigned char ∗ varDataGetter ()

  Returns the data.
- int widthGetter () const

  Returns the width of the image.
- int heightGetter () const

  Returns the height of the image.
- int channelGetter () const

  Returns the number of channels of the image.
- void setData (unsigned char ∗data)

  Replaces the data with different data.
- bool write (const std::string &write_dir, const std::string &write_filename)

  Saves the data stored in this object and saves it as an image file.

5.5.1 Detailed Description

The class saves the data of one image. It has different getters and setters that can be used to access or set the data of the image.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Image() [1/2]

```
Image::Image (
            int width,
            int height,
            int channels )
```

Creates an image with a defined width, height and number of channels. It does not preallocate the memory.

Parameters

width	Number of pixels in the x-direction.
height	Number of pixels in the y-direction.
channels	Number of channels.

5.5.2.2 Image() [2/2]

```
Image::Image (
            const char * filename )
```

It loads one image and saves the data in an 1D-Array.

Parameters

filename	Path to the image that should be used.

5.5.3 Member Function Documentation

5.5.3.1 channelGetter()

```
int Image::channelGetter ( ) const
```
Returns the number of channels of the image.

Returns

Number of channels of the image.

5.5.3.2 constDataGetter()

```
const unsigned char * Image::constDataGetter ( ) const
```
Returns the data. The data can not be changed.

Returns

The data of the image which is stored in an 1D-Array.

5.5.3.3 heightGetter()

```
int Image::heightGetter ( ) const
```
Returns the height of the image.

Returns

Height of the image.

5.5.3.4 setData()

```
void Image::setData (
            unsigned char * data )
```
Replaces the data with different data.

Parameters

data	The data that should be used to replace the old data. The new data should be stored in an 1D-Array.

5.5.3.5 varDataGetter()

```
unsigned char * Image::varDataGetter ( )
```
Returns the data.

Returns

The data of the image which is stored in an 1D-Array.

5.5.3.6 widthGetter()

```
int Image::widthGetter ( ) const
```
Returns the width of the image.

Returns

Width of the image.

5.5.3.7 write()

```
bool Image::write (
            const std::string & write_dir,
            const std::string & write_filename )
```

Saves the data stored in this object and saves it as an image file.

Parameters

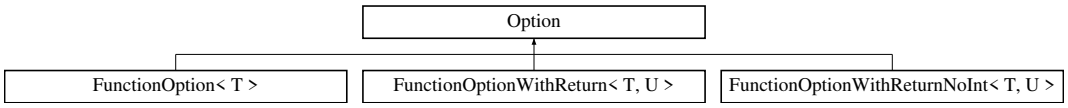
write_dir	The directory where the image should be stored.
write_filename	The filename that should be used for the image.

The documentation for this class was generated from the following files:

- src/Image.h
- src/Image.cpp

5.6 Option Struct Reference

A struct that represents an option that a User can select in the UserInterface.
`#include <UserInterface.h>`
Inheritance diagram for Option:



Public Attributes

- std::string name

5.6.1 Detailed Description

A struct that represents an option that a User can select in the UserInterface.

5.6.2 Member Data Documentation

5.6.2.1 name

`std::string Option::name`
the name of the option.
The documentation for this struct was generated from the following file:

- src/UserInterface.h

5.7 Projection Class Reference

Class with static methods to apply projections.
`#include <Projection.h>`

Static Public Member Functions

- static Image maximumIntensityProjection (Volume &volume)

  Applies the Maximum intensity projection on an volume.
- static Image minimumIntensityProjection (Volume &volume)

  Applies the Minumum intensity projection on an volume.
- static Image meanIntensityProjection (Volume &volume)

  Applies the mean intensity projection on an volume.
- static Image medianIntensityProjection (Volume &volume)

  Applies the median intensity projection on an volume.

5.7.1 Detailed Description

Class with static methods to apply projections.

5.7.2 Member Function Documentation

5.7.2.1 maximumIntensityProjection()

```
Image Projection::maximumIntensityProjection (
            Volume & volume )  [static]
```

Applies the Maximum intensity projection on an volume.

Parameters

volume	the reference of the volume that should be used for the projection.

5.7.2.2 meanIntensityProjection()

```
Image Projection::meanIntensityProjection (
            Volume & volume )  [static]
```

Applies the mean intensity projection on an volume.

Parameters

volume	the reference of the volume that should be used for the projection.

5.7.2.3 medianIntensityProjection()

```
Image Projection::medianIntensityProjection (
            Volume & volume )  [static]
```

Applies the median intensity projection on an volume.

Parameters

volume	the reference of the volume that should be used for the projection.

5.7.2.4 minimumIntensityProjection()

```
Image Projection::minimumIntensityProjection (
            Volume & volume ) [static]
```

Applies the Minumum intensity projection on an volume.

Parameters

volume	the reference of the volume that should be used for the projection.

The documentation for this class was generated from the following files:

- src/Projection.h
- src/Projection.cpp

5.8 Slice Class Reference

The class has two methods to slice a Volume into a new plane.
`#include <Slice.h>`

Static Public Member Functions

- static Image getPlaneXZ (Volume &volume, int y)

  Calcualtes the x-z plane of a volume at a fixed y position.
- static Image getPlaneYZ (Volume &volume, int x)

  Calcualtes the y-z plane of a volume at a fixed x position.

5.8.1 Detailed Description

The class has two methods to slice a Volume into a new plane.

5.8.2 Member Function Documentation

5.8.2.1 getPlaneXZ()

```
Image Slice::getPlaneXZ (
            Volume & volume,
            int y ) [static]
```

Calcualtes the x-z plane of a volume at a fixed y position.

Parameters

volume	The volume that should be used for the slicing.
y	The constant y-value that should be used for the slicing

Returns

Returns the Image from the Slicing process.

5.8.2.2 getPlaneYZ()

```
Image Slice::getPlaneYZ (
            Volume & volume,
            int x ) [static]
```

Calcualtes the y-z plane of a volume at a fixed x position.

Parameters

volume	The volume that should be used for the slicing.
y	The constant x-value that should be used for the slicing

Returns

Returns the Image from the Slicing process.

The documentation for this class was generated from the following files:

- src/Slice.h
- src/Slice.cpp

5.9 Test Class Reference

This abstract class represents one single Unit Test.
`#include <Testing.h>`

Public Member Functions

- Test (const std::string &name)

  Creates an instance of a test.
- virtual void **run** ()=0

  Runs the test. This method is pure virtual. This is why this class is abstract.
- const std::string & getName () const

  Returns the name of the test.
- void assert_true (bool condition, const std::string &name)

  Checks if the condition is true.
- void assert_false (bool condition, const std::string &name)

  Checks if the condition is false.

5.9.1 Detailed Description

This abstract class represents one single Unit Test.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Test()

```
Test::Test (
            const std::string & name )
```

Creates an instance of a test.

Parameters

name	The name of the test.

Test

5.9.3 Member Function Documentation

5.9.3.1 assert_false()

```
void Test::assert_false (
            bool condition,
            const std::string & name )
```

Checks if the condition is false.

Parameters

condition	The conditions that should be checked.
name	name of the test.

5.9.3.2 assert_true()

```
void Test::assert_true (
            bool condition,
            const std::string & name )
```

Checks if the condition is true.

Parameters

condition	The conditions that should be checked.
name	name of the test.

5.9.3.3 getName()

```
const std::string & Test::getName ( ) const
```

Returns the name of the test.

Returns

The name of the test.

The documentation for this class was generated from the following files:

- src/Testing.h
- src/Testing.cpp

5.10 TestSuite Class Reference

This class represents one Test Suite. It can contain multiple Tests.
`#include <Testing.h>`

Public Member Functions

- TestSuite (const std::string &name)

  Creates an instance of a test suite.
- const std::string & getName () const

  Returns the name of the test suite.
- void addTest (Test *test)

  Add a test to the test suite.

- void **run** ()

  Runs all the tests in the test suite.

5.10.1 Detailed Description

This class represents one Test Suite. It can contain multiple Tests.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 TestSuite()

```
TestSuite::TestSuite (
            const std::string & name )
```

Creates an instance of a test suite.

Parameters

name	The name of the test suite.

TestSuite

5.10.3 Member Function Documentation

5.10.3.1 addTest()

```
void TestSuite::addTest (
            Test * test )
```

Add a test to the test suite.

Parameters

test	The test that should be added to the Suite.

5.10.3.2 getName()

```
const std::string & TestSuite::getName ( ) const
```

Returns the name of the test suite.

Returns

The name of the test suite.

The documentation for this class was generated from the following files:

- src/Testing.h
- src/Testing.cpp

5.11 UserInterface Class Reference

The class has different methods to deal with the input of the user. It will ask the user questions and it will then call based on the user input appropriate functions.

```
#include <UserInterface.h>
```

Public Member Functions

- **UserInterface** ()

  Creates an User Interface Instance.
- void **getUserInput** ()

  This is the main function in the user Interface. This method guides the user through the whole process.
- std::string getInputPath ()

  Asks the user for the input path at the beginning.
- template<class T >
  int getSelectionFormOptions (const char *question, std::map< int, T > options)

  Gives the user different Options to choose from. The user can select one Option and the method will return this selected Option.
- std::vector< int > get2DFiltersFromUser ()

  This method asks the user which 2D Filters he wants to apply.
- template<class T >
  void applyFilters (T &data, std::map< int, FunctionOption< T > > &allFilters, std::vector< int > &filtersTo↩
  Apply)

  This function will apply all specified filter to the specified data. It will edit the data inplace.
- void applySlicing (Volume &volume, int slicer, std::string &filename)

  This function will apply one slicing on the specified volume.
- void applyProjection (Volume &volume, int projectionIdx, std::string filename, int filter3Didx=-1)

  This function will apply one projection on the specified volume.

5.11.1 Detailed Description

The class has different methods to deal with the input of the user. It will ask the user questions and it will then call based on the user input appropriate functions.

5.11.2 Member Function Documentation

5.11.2.1 applyFilters()

```
template<class T >
void UserInterface::applyFilters (
            T & data,
            std::map< int, FunctionOption< T > > & allFilters,
            std::vector< int > & filtersToApply )
```

This function will apply all specified filter to the specified data. It will edit the data inplace.

Parameters

data	The data that should be used for the filter. For example Image or Volume.
allFilters	All filters which were available for selection.
filtersToApply	Contains the filters that should be applied from `allFilters`. These are applied in the order in which they are sorted in the vector.

Template Parameters

T	Instance of the Option class that is used for the map.

5.11.2.2 applyProjection()

```
void UserInterface::applyProjection (
            Volume & volume,
            int projectionIdx,
            std::string filename,
            int filter3Didx = -1 )
```

This function will apply one projection on the specified volume.

Parameters

volume	The volume that should be used for the projection.
slicer	The projection operation that should be used for the projection. The integer corresponds the the key in the `projections` map.
filename	The filename/directory anme of the volume.
filter3Didx	The index of the 3D filter used before the projection

5.11.2.3 applySlicing()

```
void UserInterface::applySlicing (
            Volume & volume,
            int slicer,
            std::string & filename )
```

This function will apply one slicing on the specified volume.

Parameters

volume	The volume that should be used for the slicing.
slicer	The slicing operation that should be used for the slicing (x-z plane or y-z plane). The integer corresponds the the key in the `slicers` map.
filename	The filename/directory anme of the volume.

5.11.2.4 get2DFiltersFromUser()

```
std::vector< int > UserInterface::get2DFiltersFromUser ( )
```

This method asks the user which 2D Filters he wants to apply.

Returns

All the filters the user wants to apply. The vector contains integer that correspond to the key of the option in the `filters2D` map.

5.11.2.5 getInputPath()

```
std::string UserInterface::getInputPath ( )
```

Asks the user for the input path at the beginning.

Returns

The path from the user input.

5.11.2.6 getSelectionFormOptions()

```
template<class T >
int UserInterface::getSelectionFormOptions (
            const char * question,
            std::map< int, T > options )
```

Gives the user different Options to choose from. The user can select one Option and the method will return this selected Option.

Parameters

question	The question that should be displayed to the user.
options	A map that contains multiple Options. The map has integers as keys and an Instance of an Option class as a value.

Template Parameters

T	Instance of the Option class that is used for the map.

Returns

The selected option.

The documentation for this class was generated from the following files:

- src/UserInterface.h
- src/UserInterface.cpp

5.12 Util Class Reference

Utility class for sorting algorithms used in the filters and projections.

`#include <Util.h>`

Static Public Member Functions

- static unsigned char quickselect (std::vector< unsigned char > &values, int k)

  Quick-select algorithm to find the k-th smallest element in a vector of unsigned chars.
- static void mergeSort (std::vector< std::string > &strings)

  Merge sort function for strings.

5.12.1 Detailed Description

Utility class for sorting algorithms used in the filters and projections.

5.12.2 Member Function Documentation

5.12.2.1 mergeSort()

```
void Util::mergeSort (
            std::vector< std::string > & strings )  [static]
```

Merge sort function for strings.

Parameters

strings	the reference of the vector of strings to be sorted

5.12.2.2 quickselect()

```
unsigned char Util::quickselect (
            std::vector< unsigned char > & values,
            int k )  [static]
```

Quick-select algorithm to find the k-th smallest element in a vector of unsigned chars.

Parameters

values	the reference of the values to be sorted through
k	the index of the smallest element we want to find

The documentation for this class was generated from the following files:

- src/Util.h
- src/Util.cpp

5.13 Volume Class Reference

The class saves the data of multiple images. It has different getters and setters that can be used to access or set the data of the volume.

`#include <Volume.h>`

Public Member Functions

- Volume (int width, int height, int channels, int num_images)

  Creates a volume with a defined width, height, number of channels and number of images. It does not preallocate the memory.
- Volume (const char *path, int fromIdx, int toIdx, bool removeDebug=false)

  It loads all the images in the specified directory.
- const unsigned char * constDataGetter () const

  Returns the data. The data can not be changed.
- unsigned char * varDataGetter ()

  Returns the data.
- unsigned char constGetValueFromData (int x, int y, int c, int z) const

  Returns one data point in the volume given x, y, c and z.
- int widthGetter () const

  Returns the width of the Volume.
- int heightGetter () const

  Returns the height of the Volume.
- int channelsGetter () const

  Returns the number of cahnnels of the Volume.
- int numImagesGetter () const

  Returns the number of images of the Volume.
- void setData (unsigned char *data)

  Replaces the data with different data.

5.13.1 Detailed Description

The class saves the data of multiple images. It has different getters and setters that can be used to access or set the data of the volume.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 Volume() [1/2]

```
Volume::Volume (
            int width,
            int height,
            int channels,
            int num_images )
```

Creates a volume with a defined width, height, number of channels and number of images. It does not preallocate the memory.

Parameters

width	Number of pixels in the x-direction.
height	Number of pixels in the y-direction.
channels	Number of channels.
num_images	Number of images.

5.13.2.2 Volume() [2/2]

```
Volume::Volume (
            const char * path,
            int fromIdx,
            int toIdx,
            bool removeDebug = false )
```

It loads all the images in the specified directory.

Parameters

path	Path to the directory that should be used.
fromIdx	The z-coordiante from which the data should be used.
toIdx	The z-coordiante up to which the data should be used.
removeDebug	Whether the loading bar should be displayed.

5.13.3 Member Function Documentation

5.13.3.1 channelsGetter()

```
int Volume::channelsGetter ( ) const
```

Returns the number of cahnnels of the Volume.

Returns

Number of channels of the Volume.

5.13.3.2 constDataGetter()

```
const unsigned char * Volume::constDataGetter ( ) const
```

Returns the data. The data can not be changed.

Returns

The data of the volume which is stored in an 1D-Array.

5.13.3.3 constGetValueFromData()

```
unsigned char Volume::constGetValueFromData (
            int x,
            int y,
            int c,
            int z ) const
```

Returns one data point in the volume given x, y, c and z.

Returns

The data of the volume which is stored in an 1D-Array.

Parameters

x	x-coordinate of the data point.
y	y-coordinate of the data point.
c	c-coordinate of the data point.
z	z-coordinate of the data point.

5.13.3.4 heightGetter()

```
int Volume::heightGetter ( ) const
```

Returns the height of the Volume.

Returns

Height of the Volume.

5.13.3.5 numImagesGetter()

```
int Volume::numImagesGetter ( ) const
```

Returns the number of images of the Volume.

Returns

Number of images of the image.

5.13.3.6 setData()

```
void Volume::setData (
            unsigned char * data )
```

Replaces the data with different data.

Parameters

data	The data that should be used to replace the old data. The new data should be stored in an 1D-Array.

5.13.3.7 varDataGetter()

```
unsigned char * Volume::varDataGetter ( )
```

Returns the data.

Returns

The data of the volume which is stored in an 1D-Array.

5.13.3.8 widthGetter()

```
int Volume::widthGetter ( ) const
```

Returns the width of the Volume.

Returns

Width of the Volume.

The documentation for this class was generated from the following files:

- src/Volume.h
- src/Volume.cpp

Chapter 6

File Documentation

6.1 Filter.h

```
00001 /*
00002 Group name: Ziggurat
00003 Members name      |Github Username
00004 ---------------------------------------
00005 Dayou Chen        |acse-dc421
00006 Ruijia Yu         |acse-ry122
00007 Jinsong Dong      |edsml-jd622
00008 Timothy Geiger    |acse-tfg22
00009 Yue Peng          |edsml-yp22
00010 Christopher Saad  |edsml-cs1622
00011 */
00012 #pragma once
00013
00014 #include "Volume.h"
00015 #include "Image.h"
00016 #include "stb_image.h"
00017 #include "stb_image_write.h"
00018
00023 class Filter {
00024 public:
00025     // We created Methods inside the Filter class. We did not
00026     // create new Classes for Each Filter type because the
00027     // filters do not really have something in common. So
00028     // we couldn't justify inheritance.
00029     //
00030     // Every function has a second parameter call input even if the
00031     // filter does not has an additional input like kernel size. We did this
00032     // because we need functions with identical parameters in order
00033     // to store them in a map.
00034
00035
00046     static void applyGreyscaleFilter(Image& image, int input=-1);
00047
00058     static void Brightness(Image& image, int value);
00059
00068     static void applyMedianBlurFilter(Image& image, int input=-1);
00069
00078     static void applyGaussianBlurFilter(Image& image, int kernel_size=-1);
00079
00087     static void sobel(Image& image, int input=-1);
00088
00098     static void prewitt(Image& image, int input=-1);
00099
00100
00101     // 3D Filters
00102     static void gaussian3D(Volume& volume, int input=-1);
00111     static void median3D(Volume& volume, int input=-1);
00112
00113 private:
00114     static int cal_median(std::vector<int>& input_vec, int input_num);
00115     static void quick_sort(std::vector<int>& input_vec, int low, int high);
00116 };
```

6.2 Image.h

```
00001 /*
00002 Group name: Ziggurat
00003 Members name      |Github Username
00004 ---------------------------------------
```

```
00005 Dayou Chen        |acse-dc421
00006 Ruijia Yu         |acse-ry122
00007 Jinsong Dong      |edsml-jd622
00008 Timothy Geiger    |acse-tfg22
00009 Yue Peng          |edsml-yp22
00010 Christopher Saad  |edsml-cs1622
00011 */
00012 #pragma once
00013
00014 #include <string>
00015 #include "stb_image.h"
00016 #include "stb_image_write.h"
00017
00022 class Image {
00023 public:
00024
00032     Image(int width, int height, int channels);
00033
00038     Image(const char* filename);
00039
00040     ~Image();
00041
00046     const unsigned char* constDataGetter() const;
00047
00052     unsigned char* varDataGetter();
00053
00058     int widthGetter() const;
00059
00064     int heightGetter() const;
00065
00070     int channelGetter() const;
00071
00077     void setData(unsigned char* data);
00078
00085     bool write(const std::string& write_dir, const std::string& write_filename);
00086
00087 private:
00088     int width, height, channels;
00089     unsigned char* data;
00090     const char* theFilename;
00091 };
```

6.3 Projection.h

```
00001 /*
00002 Group name: Ziggurat
00003 Members name      |Github Username
00004 ---------------------------------------
00005 Dayou Chen        |acse-dc421
00006 Ruijia Yu         |acse-ry122
00007 Jinsong Dong      |edsml-jd622
00008 Timothy Geiger    |acse-tfg22
00009 Yue Peng          |edsml-yp22
00010 Christopher Saad  |edsml-cs1622
00011 */
00012 #pragma once
00013
00014 #include "Volume.h"
00015 #include "Image.h"
00016 #include <functional>
00017
00021 class Projection
00022 {
00023 public:
00030     static Image maximumIntensityProjection(Volume& volume);
00031
00038     static Image minimumIntensityProjection(Volume& volume);
00039
00046     static Image meanIntensityProjection(Volume& volume);
00047
00054     static Image medianIntensityProjection(Volume& volume);
00055
00056
00057 private:
00058     // static Image projection(Volume& volume, std::function<unsigned char(std::vector<unsigned
      char>)> proj_function);
00059     static Image intensityProjection(Volume& volume, std::function<unsigned char(Volume&, int, int,
      int, int)> projectionFunction);
00060     static unsigned char maxPixelProjection(Volume& volume, int i, int j, int c, int z);
00061     static unsigned char minPixelProjection(Volume& volume, int i, int j, int c, int z);
00062     static unsigned char meanPixelProjection(Volume& volume, int i, int j, int c, int z);
00063     static unsigned char medianPixelProjection(Volume& volume, int i, int j, int c, int z);
00064 };
```

6.4 Slice.h

```
00001 /*
00002 Group name: Ziggurat
00003 Members name      |Github Username
00004 -------------------------------------
00005 Dayou Chen        |acse-dc421
00006 Ruijia Yu         |acse-ry122
00007 Jinsong Dong      |edsml-jd622
00008 Timothy Geiger    |acse-tfg22
00009 Yue Peng          |edsml-yp22
00010 Christopher Saad  |edsml-cs1622
00011 */
00012 #pragma once
00013
00014 #include "Image.h"
00015 #include "Volume.h"
00016 #include <iostream>
00017
00022 class Slice {
00023 public:
00024
00031     static Image getPlaneXZ(Volume& volume, int y);
00032
00039     static Image getPlaneYZ(Volume& volume, int x);
00040 };
```

6.5 Testing.h

```
00001 /*
00002 Group name: Ziggurat
00003 Members name      |Github Username
00004 -------------------------------------
00005 Dayou Chen        |acse-dc421
00006 Ruijia Yu         |acse-ry122
00007 Jinsong Dong      |edsml-jd622
00008 Timothy Geiger    |acse-tfg22
00009 Yue Peng          |edsml-yp22
00010 Christopher Saad  |edsml-cs1622
00011 */
00012 #include <string>
00013 #include <vector>
00014
00018 class Test {
00019 public:
00020
00025     Test(const std::string& name);
00026
00031     virtual void run() = 0;
00032
00037     const std::string& getName() const;
00038
00044     void assert_true(bool condition, const std::string& name);
00045
00051     void assert_false(bool condition, const std::string& name);
00052
00053 private:
00054     std::string name;
00055 };
00056
00057
00061 class TestSuite {
00062 public:
00063
00068     TestSuite(const std::string& name);
00069
00070     ~TestSuite();
00071
00076     const std::string& getName() const;
00077
00082     void addTest(Test* test);
00083
00087     void run();
00088
00089 private:
00090     std::string name;
00091     std::vector<Test*> tests;
00092 };
```

6.6 UserInterface.h

```
00001 /*
```

```
00002 Group name: Ziggurat
00003 Members name      |Github Username
00004 ---------------------------------
00005 Dayou Chen        |acse-dc421
00006 Ruijia Yu         |acse-ry122
00007 Jinsong Dong      |edsml-jd622
00008 Timothy Geiger    |acse-tfg22
00009 Yue Peng          |edsml-yp22
00010 Christopher Saad  |edsml-cs1622
00011 */
00012 #pragma once
00013 #include <iostream>
00014 #include <string>
00015 #include <vector>
00016 #include <map>
00017
00018 // to have functions as variables
00019 #include <functional>
00020
00021 // for clearing the screen
00022 #include <cstdlib>
00023
00024 // for creating a folder
00025 #include <sys/stat.h>
00026
00027 #include "Filter.h"
00028 #include "Volume.h"
00029 #include "Slice.h"
00030 #include "Projection.h"
00031
00036 struct Option {
00037     std::string name;
00038 };
00039
00046 template <class T>
00047 struct FunctionOption : Option {
00048     std::function<void(T&, int)> function;
00052     std::string additionalString = "";
00055     int input;
00056 };
00057
00067 template <class T, class U>
00068 struct FunctionOptionWithReturn : Option {
00069     std::function<U(T&, int)> function;
00072     std::string additionalString = "";
00075     int input;
00076 };
00077
00087 template <class T, class U>
00088 struct FunctionOptionWithReturnNoInt : Option {
00089     std::function<U(T&)> function;
00092     std::string additionalString = "";
00095     int input;
00096 };
00097
00098
00104 class UserInterface {
00105 public:
00106
00110     UserInterface();
00111
00112     ~UserInterface();
00113
00118     void getUserInput();
00119
00125     std::string getInputPath();
00126
00137     template<class T>
00138     int getSelectionFormOptions(const char* question, std::map<int, T> options);
00139
00145     std::vector<int> get2DFiltersFromUser();
00146
00158     template <class T>
00159     void applyFilters(T& data, std::map<int, FunctionOption<T>>& allFilters, std::vector<int>&
      filtersToApply);
00160
00169     void applySlicing(Volume& volume, int slicer, std::string& filename);
00170
00179     void applyProjection(Volume& volume, int projectionIdx, std::string filename, int filter3Didx=-1);
00180
00181 private:
00182     std::map<int, FunctionOption<Image» filters2D = {
00183         {1, {"greyscale", Filter::applyGreyscaleFilter}},
00184         {2, {"brightness", Filter::Brightness, "Please input a brightness number: "}},
00185         {3, {"median-blur", Filter::applyMedianBlurFilter, "Please input the kernel size for the
      median blur: "}},
00186         {4, {"gaussian-blur", Filter::applyGaussianBlurFilter, "Please input the kernel size for the
```

```
    gaussian blur: "}},
00187         {5, {"sobel", Filter::sobel}},
00188         {6, {"prewitt", Filter::prewitt}}
00189     };
00190 
00191     std::map<int, FunctionOptionWithReturn<Volume, Image» slicers = {
00192         {1, {"x-z plane", Slice::getPlaneXZ, "y"}},
00193         {2, {"y-z plane", Slice::getPlaneYZ, "x"}},
00194     };
00195 
00196     std::map<int, FunctionOptionWithReturnNoInt<Volume, Image» projections = {
00197         {1, {"Maximum intensity projection (MIP)", Projection::maximumIntensityProjection}},
00198         {2, {"Minimum intensity projection (MinIP)", Projection::minimumIntensityProjection}},
00199         {3, {"Mean - Average intensity projection (AIP)", Projection::meanIntensityProjection}},
00200         {4, {"Median - Average intensity projection (AIP)", Projection::medianIntensityProjection}}
00201     };
00202 
00203     std::map<int, FunctionOption<Volume» filters3D = {
00204         {1, {"3D Gaussian", Filter::gaussian3D, "Please input a kernel size for the 3D Gaussian blur
    filter: "}},
00205         {2, {"3D Median", Filter::median3D, "Please input a kernel size for the 3D Median blur filter:
    "}}
00206     };
00207 };
```

6.7 Util.h

```
00001 /*
00002 Group name: Ziggurat
00003 Members name      |Github Username
00004 ---------------------------------------
00005 Dayou Chen        |acse-dc421
00006 Ruijia Yu         |acse-ry122
00007 Jinsong Dong      |edsml-jd622
00008 Timothy Geiger    |acse-tfg22
00009 Yue Peng          |edsml-yp22
00010 Christopher Saad  |edsml-cs1622
00011 */
00012 #include <vector>
00013 #include <string>
00014 
00018 class Util {
00019 public:
00026     static unsigned char quickselect(std::vector<unsigned char>& values, int k);
00032     static void mergeSort(std::vector<std::string>& strings);
00033 private:
00034     static int partition(std::vector<unsigned char>& values, int left, int right);
00035     static void swap(std::vector<unsigned char>& values, int i, int j);
00036     static void mergeHelper(std::vector<std::string>& strings, int start, int mid, int end);
00037 };
```

6.8 Volume.h

```
00001 /*
00002 Group name: Ziggurat
00003 Members name      |Github Username
00004 ---------------------------------------
00005 Dayou Chen        |acse-dc421
00006 Ruijia Yu         |acse-ry122
00007 Jinsong Dong      |edsml-jd622
00008 Timothy Geiger    |acse-tfg22
00009 Yue Peng          |edsml-yp22
00010 Christopher Saad  |edsml-cs1622
00011 */
00012 #pragma once
00013 
00014 #include <iostream>
00015 #include <string>
00016 #include <filesystem>
00017 #include <fstream>
00018 #include <vector>
00019 #include "stb_image.h"
00020 #include "stb_image_write.h"
00021 
00027 class Volume {
00028 public:
00029 
00038     Volume(int width, int height, int channels, int num_images);
00039 
00047     Volume(const char* path, int fromIdx, int toIdx, bool removeDebug=false);
00048 
00049     ~Volume();
```

```
00050
00055     const unsigned char* constDataGetter() const;
00056
00061     unsigned char* varDataGetter();
00062
00071     unsigned char constGetValueFromData(int x, int y, int c, int z) const;
00072
00077     int widthGetter() const;
00078
00083     int heightGetter() const;
00084
00089     int channelsGetter() const;
00090
00095     int numImagesGetter() const;
00096
00102     void setData(unsigned char* data);
00103
00104 private:
00105     int width, height, channels, num_images;
00106     unsigned char* data;
00107     const char* path;
00108 };
```