Image Filters, Projections and Slices

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 Blur2D< WORK, STORE > Class Template Reference

```
#include <Filter.hpp>
```

Inheritance diagram for Blur2D< WORK, STORE >:

Collaboration diagram for Blur2D< WORK, STORE >:

Public Member Functions

- Blur2D (const BlurKernel &, const int &=3, const WORK &=1.)
- WORK applyPerElement (const Image< STORE > &, const int &, const int &, const int &) const

Protected Attributes

- BlurKernel kernel

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Blur2D()

```
template<typename WORK , typename STORE >
Blur2D< WORK, STORE >::Blur2D (
             const BlurKernel & arg_kernel,
             const int & arg_kernel_size = 3,
             const WORK & stddev = 1.  )
```

4.1.2 Member Function Documentation

4.1.2.1 applyPerElement()

```
template<typename WORK , typename STORE >
WORK Blur2D< WORK, STORE >::applyPerElement (
             const Image< STORE > & original_image,
             const int & w,
             const int & h,
             const int & c ) const  [virtual]
```

Reimplemented from Filter2D< WORK, STORE >.

Here is the call graph for this function:

4.1.3 Member Data Documentation

4.1.3.1 kernel

```
template<typename WORK , typename STORE >
BlurKernel Blur2D< WORK, STORE >::kernel  [protected]
```

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

- src/Filter.hpp
- src/Filter.cpp

4.2 Blur3D< WORK, STORE > Class Template Reference

```
#include <Filter.hpp>
```

Inheritance diagram for Blur3D< WORK, STORE >:

Collaboration diagram for Blur3D< WORK, STORE >:

Public Member Functions

- Blur3D (const BlurKernel &, const int &=3, const WORK &=1.)
- WORK applyPerElement (const Volume< STORE > &, const int &, const int &, const int &, const int &) const

Protected Attributes

- BlurKernel kernel

4.2.1 Constructor & Destructor Documentation

4.2.1.1 Blur3D()

```
template<typename WORK , typename STORE >
Blur3D< WORK, STORE >::Blur3D (
             const BlurKernel & arg_kernel,
             const int & arg_kernel_size = 3,
             const WORK & stddev = 1.  )
```

4.2.2 Member Function Documentation

4.2.2.1 applyPerElement()

```
template<typename WORK , typename STORE >
WORK Blur3D< WORK, STORE >::applyPerElement (
             const Volume< STORE > & original_volume,
             const int & d,
             const int & w,
             const int & h,
             const int & c ) const  [virtual]
```

Reimplemented from Filter3D< WORK, STORE >.

Here is the call graph for this function:

4.2.3 Member Data Documentation

4.2.3.1 kernel

```
template<typename WORK , typename STORE >
BlurKernel Blur3D< WORK, STORE >::kernel  [protected]
```

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

- src/Filter.hpp
- src/Filter.cpp

4.3 Brightness2D< WORK, STORE > Class Template Reference

```
#include <Filter.hpp>
```

Inheritance diagram for Brightness2D< WORK, STORE >:

Collaboration diagram for Brightness2D< WORK, STORE >:

Public Member Functions

- Brightness2D (const WORK &)
- WORK applyPerElement (const Image< STORE > &, const int &, const int &, const int &) const

Protected Attributes

- WORK brightness

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Brightness2D()

```
template<typename WORK , typename STORE >
Brightness2D< WORK, STORE >::Brightness2D (
             const WORK & arg_brightness )
```

4.3.2 Member Function Documentation

4.3.2.1 applyPerElement()

```
template<typename WORK , typename STORE >
WORK Brightness2D< WORK, STORE >::applyPerElement (
             const Image< STORE > & original_image,
             const int & w,
             const int & h,
             const int & c ) const  [virtual]
```

Reimplemented from Filter2D< WORK, STORE >.

Here is the call graph for this function:

4.3.3 Member Data Documentation

4.3.3.1 brightness

```
template<typename WORK , typename STORE >
WORK Brightness2D< WORK, STORE >::brightness  [protected]
```

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

- src/Filter.hpp
- src/Filter.cpp

4.4 ColourBalance2D< WORK, STORE > Class Template Reference

```
#include <Filter.hpp>
```

Inheritance diagram for ColourBalance2D< WORK, STORE >:

Collaboration diagram for ColourBalance2D< WORK, STORE >:

Public Member Functions

- ColourBalance2D (const std::vector< WORK > &)
- WORK applyPerElement (const Image< STORE > &, const int &, const int &, const int &) const
- Image< STORE > apply (const Image< STORE > &) const

Protected Attributes

- std::vector< WORK > channel_coefficients

4.4.1 Constructor & Destructor Documentation

4.4.1.1 ColourBalance2D()

```
template<typename WORK , typename STORE >
ColourBalance2D< WORK, STORE >::ColourBalance2D (
            const std::vector< WORK > & coefficients )
```

4.4.2 Member Function Documentation

4.4.2.1 apply()

```
template<typename WORK , typename STORE >
Image< STORE > ColourBalance2D< WORK, STORE >::apply (
            const Image< STORE > & original_image ) const
```

Here is the call graph for this function: Here is the caller graph for this function:

4.4.2.2 applyPerElement()

```
template<typename WORK , typename STORE >
WORK ColourBalance2D< WORK, STORE >::applyPerElement (
            const Image< STORE > & original_image,
            const int & w,
            const int & h,
            const int & c ) const  [virtual]
```

Reimplemented from Filter2D< WORK, STORE >.

Here is the call graph for this function:

4.4.3 Member Data Documentation

4.4.3.1 channel_coefficients

```
template<typename WORK , typename STORE >
std::vector<WORK> ColourBalance2D< WORK, STORE >::channel_coefficients  [protected]
```

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

- src/Filter.hpp
- src/Filter.cpp

4.5 Conv2D< WORK, STORE > Class Template Reference

```
#include <Filter.hpp>
```

Inheritance diagram for Conv2D< WORK, STORE >:

Collaboration diagram for Conv2D< WORK, STORE >:

Public Member Functions

- Conv2D (const int &)

Protected Attributes

- int kernel_size
- std::vector< int > kernel_range
- std::vector< WORK > kernel_parameters

4.5.1 Constructor & Destructor Documentation

4.5.1.1 Conv2D()

```
template<typename WORK , typename STORE >
Conv2D< WORK, STORE >::Conv2D (
             const int & arg_kernel_size )
```

4.5.2 Member Data Documentation

4.5.2.1 kernel_parameters

```
template<typename WORK , typename STORE >
std::vector<WORK> Conv2D< WORK, STORE >::kernel_parameters  [protected]
```

4.5.2.2 kernel_range

```
template<typename WORK , typename STORE >
std::vector<int> Conv2D< WORK, STORE >::kernel_range  [protected]
```

4.5.2.3 kernel_size

```
template<typename WORK , typename STORE >
int Conv2D< WORK, STORE >::kernel_size  [protected]
```

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

- src/Filter.hpp
- src/Filter.cpp

4.6 Conv3D< WORK, STORE > Class Template Reference

```
#include <Filter.hpp>
```

Inheritance diagram for Conv3D< WORK, STORE >:

Collaboration diagram for Conv3D< WORK, STORE >:

Public Member Functions

- Conv3D (const int &)

Protected Attributes

- int kernel_size
- std::vector< int > kernel_range
- std::vector< WORK > kernel_parameters

4.6.1 Constructor & Destructor Documentation

4.6.1.1 Conv3D()

```
template<typename WORK , typename STORE >
Conv3D< WORK, STORE >::Conv3D (
             const int & arg_kernel_size )
```

4.6.2 Member Data Documentation

4.6.2.1 kernel_parameters

```
template<typename WORK , typename STORE >
std::vector<WORK> Conv3D< WORK, STORE >::kernel_parameters  [protected]
```

4.6.2.2 kernel_range

```
template<typename WORK , typename STORE >
std::vector<int> Conv3D< WORK, STORE >::kernel_range  [protected]
```

4.6.2.3 kernel_size

```
template<typename WORK , typename STORE >
int Conv3D< WORK, STORE >::kernel_size  [protected]
```

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

- src/Filter.hpp
- src/Filter.cpp

4.7 Edge2D< WORK, STORE > Class Template Reference

```
#include <Filter.hpp>
```

Inheritance diagram for Edge2D< WORK, STORE >:

Collaboration diagram for Edge2D< WORK, STORE >:

Public Member Functions

- Edge2D (const EdgeKernel &)
- WORK applyPerElement (const Image< STORE > &, const int &, const int &, const int &) const
- Image< STORE > apply (const Image< STORE > &) const

Protected Attributes

- EdgeKernel kernel

4.7.1 Constructor & Destructor Documentation

4.7.1.1 Edge2D()

```
template<typename WORK , typename STORE >
Edge2D< WORK, STORE >::Edge2D (
            const EdgeKernel & arg_kernel )
```

4.7.2 Member Function Documentation

4.7.2.1 apply()

```
template<typename WORK , typename STORE >
Image< STORE > Edge2D< WORK, STORE >::apply (
            const Image< STORE > & original_image ) const
```

Here is the call graph for this function: Here is the caller graph for this function:

4.7.2.2 applyPerElement()

```
template<typename WORK , typename STORE >
WORK Edge2D< WORK, STORE >::applyPerElement (
             const Image< STORE > & original_image,
             const int & w,
             const int & h,
             const int & c ) const  [virtual]
```

Reimplemented from Filter2D< WORK, STORE >.

Here is the call graph for this function:

4.7.3 Member Data Documentation

4.7.3.1 kernel

```
template<typename WORK , typename STORE >
EdgeKernel Edge2D< WORK, STORE >::kernel  [protected]
```

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

- src/Filter.hpp
- src/Filter.cpp

4.8 Filter2D< WORK, STORE > Class Template Reference

```
#include <Filter.hpp>
```

Inheritance diagram for Filter2D< WORK, STORE >:

Collaboration diagram for Filter2D< WORK, STORE >:

Public Member Functions

- virtual WORK applyPerElement (const Image< STORE > &, const int &, const int &, const int &) const
- virtual Image< STORE > apply (const Image< STORE > &, const int &=-1, const std::vector< int > &={-1, -1}, const std::vector< int > &={-1, -1}) const

4.8.1 Member Function Documentation

4.8.1.1 apply()

```
template<typename WORK , typename STORE >
Image< STORE > Filter2D< WORK, STORE >::apply (
            const Image< STORE > & original_image,
            const int & arg_channels = -1,
            const std::vector< int > & arg_width_range = {-1, -1},
            const std::vector< int > & arg_height_range = {-1, -1} ) const  [virtual]
```

Reimplemented in HistogramEqualisation2D< WORK, STORE >.

Here is the call graph for this function: Here is the caller graph for this function:

4.8.1.2 applyPerElement()

```
template<typename WORK , typename STORE >
WORK Filter2D< WORK, STORE >::applyPerElement (
            const Image< STORE > & original_image,
            const int & w,
            const int & h,
            const int & c ) const  [virtual]
```

Reimplemented in Edge2D< WORK, STORE >, MedianBlur2D< WORK, STORE >, Blur2D< WORK, STORE >, Brightness2D< WORK, STORE >, and ColourBalance2D< WORK, STORE >.

Here is the call graph for this function:

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

- src/Filter.hpp
- src/Filter.cpp

4.9 Filter3D< WORK, STORE > Class Template Reference

```
#include <Filter.hpp>
```

Inheritance diagram for Filter3D< WORK, STORE >:

Collaboration diagram for Filter3D< WORK, STORE >:

Public Member Functions

- virtual WORK applyPerElement (const Volume< STORE > &, const int &, const int &, const int &, const int &) const
- virtual Volume< STORE > apply (const Volume< STORE > &, const int &=-1, const std::vector< int > &={-1, -1}, const std::vector< int > &={-1, -1}, const std::vector< int > &={-1, -1}) const

4.9.1 Member Function Documentation

4.9.1.1 apply()

```
template<typename WORK , typename STORE >
Volume< STORE > Filter3D< WORK, STORE >::apply (
            const Volume< STORE > & original_volume,
            const int & arg_channels = -1,
            const std::vector< int > & arg_depth_range = {-1, -1},
            const std::vector< int > & arg_width_range = {-1, -1},
            const std::vector< int > & arg_height_range = {-1, -1} ) const  [virtual]
```

Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.2 applyPerElement()

```
template<typename WORK , typename STORE >
WORK Filter3D< WORK, STORE >::applyPerElement (
            const Volume< STORE > & original_volume,
            const int & d,
            const int & w,
            const int & h,
            const int & c ) const  [virtual]
```

Reimplemented in MedianBlur3D< WORK, STORE >, and Blur3D< WORK, STORE >.

Here is the call graph for this function:

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

- src/Filter.hpp
- src/Filter.cpp

4.10 Grayscale2D< WORK, STORE > Class Template Reference

```
#include <Filter.hpp>
```

Inheritance diagram for Grayscale2D< WORK, STORE >:

Collaboration diagram for Grayscale2D< WORK, STORE >:

Public Member Functions

- Grayscale2D (const std::vector< int > &={0, 1, 2}, const std::vector< WORK > &={0.2126, 0.7152, 0.0722})
- WORK applyPerElement (const Image< STORE > &, const int &, const int &) const
- Image< STORE > apply (const Image< STORE > &) const

Protected Attributes

- std::vector< int > channel_indices
- std::vector< WORK > channel_coefficients

4.10.1 Constructor & Destructor Documentation

4.10.1.1 Grayscale2D()

```
template<typename WORK , typename STORE >
Grayscale2D< WORK, STORE >::Grayscale2D (
             const std::vector< int > & indices = {0, 1, 2},
             const std::vector< WORK > & coefficients = {0.2126, 0.7152, 0.0722} )
```

4.10.2 Member Function Documentation

4.10.2.1 apply()

```
template<typename WORK , typename STORE >
Image< STORE > Grayscale2D< WORK, STORE >::apply (
             const Image< STORE > & original_image ) const
```

Here is the call graph for this function: Here is the caller graph for this function:

4.10.2.2 applyPerElement()

```
template<typename WORK , typename STORE >
WORK Grayscale2D< WORK, STORE >::applyPerElement (
             const Image< STORE > & original_image,
             const int & w,
             const int & h ) const
```

Here is the call graph for this function:

4.10.3 Member Data Documentation

4.10.3.1 channel_coefficients

```
template<typename WORK , typename STORE >
std::vector<WORK> Grayscale2D< WORK, STORE >::channel_coefficients  [protected]
```

4.10.3.2 channel_indices

```
template<typename WORK , typename STORE >
std::vector<int> Grayscale2D< WORK, STORE >::channel_indices  [protected]
```

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

- src/Filter.hpp
- src/Filter.cpp

4.11 HistogramEqualisation2D< WORK, STORE > Class Template Reference

```
#include <Filter.hpp>
```

Inheritance diagram for HistogramEqualisation2D< WORK, STORE >:

Collaboration diagram for HistogramEqualisation2D< WORK, STORE >:

Public Member Functions

- Image< STORE > apply (const Image< STORE > &, const int &=-1, const std::vector< int > &={-1, -1}, const std::vector< int > &={-1, -1}) const

4.11.1 Member Function Documentation

4.11.1.1 apply()

```
template<typename WORK , typename STORE >
Image< STORE > HistogramEqualisation2D< WORK, STORE >::apply (
             const Image< STORE > & original_image,
             const int & arg_channels = -1,
             const std::vector< int > & arg_width_range = {-1, -1},
             const std::vector< int > & arg_height_range = {-1, -1} ) const  [virtual]
```

Reimplemented from Filter2D< WORK, STORE >.

Here is the call graph for this function: Here is the caller graph for this function:

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

- src/Filter.hpp
- src/Filter.cpp

4.12 Image< DATA > Class Template Reference

Class for representing and manipulating images.

```
#include <Image.hpp>
```

Collaboration diagram for Image< DATA >:

Public Member Functions

- Image (const char ∗)

    Constructs an Image object from a file.
- Image (const int &, const int &, const int &)

    Constructs an Image object with user-allocated memory.
- Image & operator= (const Image< DATA > &)
- ∼Image ()
- const int & getWidth () const

    Destructor for the Image object.
- const int & getHeight () const
- const int & getChannels () const
- void setPixel (const DATA &, const int &, const int &, const int &=0)

    Sets the pixel value at the specified position.
- const DATA & getPixel (const int &, const int &, const int &=0, const PaddingType &=PaddingType::extend, const DATA &=0) const
- void write (const char ∗) const

Static Public Member Functions

- static const int reflect_into_domain (const int &, const int &, const int &)
- static const int wrap_into_domain (const int &, const int &, const int &)
- static void check_range_assign (std::vector< int > &, const std::vector< int > &, const int &, const int &)

Protected Attributes

- ImageAllocation image_allocation
- int width
- int height
- int channels
- DATA ∗ pixels

4.12.1 Detailed Description

template<typename DATA = unsigned char>
class Image< DATA >

Class for representing and manipulating images.

Template Parameters

DATA	The data type of the image pixels.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 Image() [1/2]

```
template<typename DATA >
Image< DATA >::Image (
             const char * image_filename )
```

Constructs an Image object from a file.

Parameters

in	*filename*	The filename of the image file to read.

4.12.2.2 Image() [2/2]

```
template<typename DATA >
Image< DATA >::Image (
             const int & w,
             const int & h,
             const int & c )
```

Constructs an Image object with user-allocated memory.

Parameters

in	*width*	The width of the image.
in	*height*	The height of the image.
in	*channels*	The number of channels of the image.

4.12.2.3 ∼Image()

```
template<typename DATA >
Image< DATA >::∼Image
```

4.12.3 Member Function Documentation

4.12.3.1 check_range_assign()

```
template<typename DATA >
void Image< DATA >::check_range_assign (
             std::vector< int > & output_vector,
             const std::vector< int > & input_vector,
             const int & range_min,
             const int & range_max )  [static]
```

Here is the caller graph for this function:

4.12.3.2 getChannels()

```
template<typename DATA >
const int & Image< DATA >::getChannels
```

Here is the caller graph for this function:

4.12.3.3 getHeight()

```
template<typename DATA >
const int & Image< DATA >::getHeight
```

Here is the caller graph for this function:

4.12.3.4 getPixel()

```
template<typename DATA >
const DATA & Image< DATA >::getPixel (
             const int & x,
             const int & y,
             const int & ch = 0,
             const PaddingType & pad = PaddingType::extend,
             const DATA & constant_pad_value = 0 ) const
```

Here is the caller graph for this function:

4.12.3.5 getWidth()

```
template<typename DATA >
const int & Image< DATA >::getWidth
```

Destructor for the Image object.

Here is the caller graph for this function:

4.12.3.6 operator=()

```
template<typename DATA >
Image< DATA > & Image< DATA >::operator= (
             const Image< DATA > & other_image )
```

4.12.3.7 reflect_into_domain()

```
template<typename DATA >
const int Image< DATA >::reflect_into_domain (
             const int & num,
             const int & num_min,
             const int & num_max ) [static]
```

Here is the caller graph for this function:

4.12.3.8 setPixel()

```
template<typename DATA >
void Image< DATA >::setPixel (
             const DATA & value,
             const int & x,
             const int & y,
             const int & ch = 0 )
```

Sets the pixel value at the specified position.

Parameters

in	*value*	The value to set the pixel to.
in	*row*	The row index of the pixel.
in	*col*	The column index of the pixel.
in	*channel*	The channel index of the pixel.

Here is the caller graph for this function:

4.12.3.9 wrap_into_domain()

```
template<typename DATA >
const int Image< DATA >::wrap_into_domain (
             const int & num,
             const int & num_min,
             const int & num_max ) [static]
```

Here is the caller graph for this function:

4.12.3.10 write()

```
template<typename DATA >
void Image< DATA >::write (
            const char * image_filename ) const
```

4.12.4 Member Data Documentation

4.12.4.1 channels

```
template<typename DATA = unsigned char>
int Image< DATA >::channels  [protected]
```

4.12.4.2 height

```
template<typename DATA = unsigned char>
int Image< DATA >::height  [protected]
```

4.12.4.3 image_allocation

```
template<typename DATA = unsigned char>
ImageAllocation Image< DATA >::image_allocation  [protected]
```

4.12.4.4 pixels

```
template<typename DATA = unsigned char>
DATA* Image< DATA >::pixels  [protected]
```

4.12.4.5 width

```
template<typename DATA = unsigned char>
int Image< DATA >::width  [protected]
```

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

- src/Image.hpp
- src/Image.cpp

4.13 MedianBlur2D< WORK, STORE > Class Template Reference

`#include <Filter.hpp>`

Inheritance diagram for MedianBlur2D< WORK, STORE >:

Collaboration diagram for MedianBlur2D< WORK, STORE >:

Public Member Functions

- MedianBlur2D (const int &=3)
- WORK applyPerElement (const Image< STORE > &, const int &, const int &, const int &) const

Additional Inherited Members

4.13.1 Constructor & Destructor Documentation

4.13.1.1 MedianBlur2D()

```
template<typename WORK , typename STORE >
MedianBlur2D< WORK, STORE >::MedianBlur2D (
             const int & arg_kernel_size = 3 )
```

4.13.2 Member Function Documentation

4.13.2.1 applyPerElement()

```
template<typename WORK , typename STORE >
WORK MedianBlur2D< WORK, STORE >::applyPerElement (
             const Image< STORE > & original_image,
             const int & w,
             const int & h,
             const int & c ) const  [virtual]
```

Reimplemented from Filter2D< WORK, STORE >.

Here is the call graph for this function:

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

- src/Filter.hpp
- src/Filter.cpp

4.14 MedianBlur3D< WORK, STORE > Class Template Reference

```
#include <Filter.hpp>
```

Inheritance diagram for MedianBlur3D< WORK, STORE >:

Collaboration diagram for MedianBlur3D< WORK, STORE >:

Public Member Functions

- MedianBlur3D (const int &=3)
- WORK applyPerElement (const Volume< STORE > &, const int &, const int &, const int &, const int &) const

Additional Inherited Members

4.14.1 Constructor & Destructor Documentation

4.14.1.1 MedianBlur3D()

```
template<typename WORK , typename STORE >
MedianBlur3D< WORK, STORE >::MedianBlur3D (
             const int & arg_kernel_size = 3 )
```

4.14.2 Member Function Documentation

4.14.2.1 applyPerElement()

```
template<typename WORK , typename STORE >
WORK MedianBlur3D< WORK, STORE >::applyPerElement (
             const Volume< STORE > & original_volume,
             const int & d,
             const int & w,
             const int & h,
             const int & c ) const  [virtual]
```

Reimplemented from Filter3D< WORK, STORE >.

Here is the call graph for this function:

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

- src/Filter.hpp
- src/Filter.cpp

4.15 Projection< WORK, STORE > Class Template Reference

Class for performing projections of volumetric data onto 2D images.

```
#include <Projection.hpp>
```

Collaboration diagram for Projection< WORK, STORE >:

Public Member Functions

- Image< STORE > take (const Volume< STORE > &, const Intensity &, const Axis &=Axis::elevation, const std::vector< int > &={-1, -1})

  Computes a 2D projection of volumetric data.

4.15.1 Detailed Description

template<typename WORK, typename STORE>
class Projection< WORK, STORE >

Class for performing projections of volumetric data onto 2D images.

Template Parameters

WORK	The working precision data type of the projection.
STORE	The storage precision data type of the projection.

4.15.2 Member Function Documentation

4.15.2.1 take()

```
template<typename WORK , typename STORE >
Image< STORE > Projection< WORK, STORE >::take (
            const Volume< STORE > & original_volume,
            const Intensity & projection_intensity,
            const Axis & projection_axis = Axis::elevation,
            const std::vector< int > & slice_limits = {-1, -1} )
```

Computes a 2D projection of volumetric data.

This method computes a 2D projection of the volumetric data along the specified axis, and with the specified intensity value.

Parameters

in	*volume*	The volumetric data to project.
in	*intensity*	The type of intensity value to use for the projection.
in	*axis*	The axis along which to project the data.
in	*range*	The range of the projection along the specified axis.

Returns

The 2D image containing the projection.

Here is the call graph for this function: Here is the caller graph for this function:

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

- src/Projection.hpp
- src/Projection.cpp

4.16 Quickselect< T > Class Template Reference

Class for performing quickselect and median selection algorithms on a vector.

```
#include <Quickselect.hpp>
```

Collaboration diagram for Quickselect< T >:

Static Public Member Functions

- static std::size_t partition (std::vector< T > &, const std::size_t &, const std::size_t &)

  Partitions a vector around a pivot element.
- static T quickselect (std::vector< T > &, const std::size_t &, const std::size_t &, const std::size_t &)

  Performs the quickselect algorithm on a vector.
- static T median (std::vector< T > &)

  Computes the median of a vector using the quickselect algorithm.

4.16.1 Detailed Description

template<typename T>
class Quickselect< T >

Class for performing quickselect and median selection algorithms on a vector.

Template Parameters

T	The data type of the elements in the vector.

4.16.2 Member Function Documentation

4.16.2.1 median()

```
template<typename T >
T Quickselect< T >::median (
            std::vector< T > & vec )  [static]
```

Computes the median of a vector using the quickselect algorithm.

This method returns the median element of the given vector.

Parameters

in,out	*vec*	The vector to compute the median of.

Returns

The median element of the vector.

Here is the caller graph for this function:

4.16.2.2 partition()

```
template<typename T >
std::size_t Quickselect< T >::partition (
            std::vector< T > & vec,
            const std::size_t & begin,
            const std::size_t & end )  [static]
```

Partitions a vector around a pivot element.

This method partitions the given vector such that all elements less than the pivot element appear before it, and all elements greater than or equal to it appear after it.

Parameters

in,out	*vec*	The vector to partition.
in	*lo*	The starting index of the partition.
in	*hi*	The ending index of the partition.

Returns

The index of the pivot element after partitioning.

4.16.2.3 quickselect()

```
template<typename T >
T Quickselect< T >::quickselect (
             std::vector< T > & vec,
             const std::size_t & begin,
             const std::size_t & end,
             const std::size_t & index )   [static]
```

Performs the quickselect algorithm on a vector.

This method returns the k-th smallest element of the given vector after partitioning it.

Parameters

`in,out`	*vec*	The vector to perform quickselect on.
`in`	*lo*	The starting index of the quickselect.
`in`	*hi*	The ending index of the quickselect.
`in`	*k*	The index of the desired element after partitioning.

Returns

The k-th smallest element of the vector after partitioning.

Here is the call graph for this function: Here is the caller graph for this function:

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

- src/Quickselect.hpp
- src/Quickselect.cpp

4.17 Slice< WORK, STORE > Class Template Reference

Class for extracting 2D slices from 3D volumetric data.

`#include <Slice.hpp>`

Collaboration diagram for Slice< WORK, STORE >:

Public Member Functions

- Image< STORE > take (const Volume< STORE > &, const Axis &, const int &)

  Extracts a 2D slice from 3D volumetric data along a specified axis.

4.17.1 Detailed Description

template<typename WORK, typename STORE>
class Slice< WORK, STORE >

Class for extracting 2D slices from 3D volumetric data.

Template Parameters

WORK	The working precision data type of the slice.
STORE	The storage precision data type of the slice.

4.17.2 Member Function Documentation

4.17.2.1 take()

```
template<typename WORK , typename STORE >
Image< STORE > Slice< WORK, STORE >::take (
             const Volume< STORE > & original_volume,
             const Axis & slice_axis,
             const int & slice_location )
```

Extracts a 2D slice from 3D volumetric data along a specified axis.

This method extracts a 2D slice from the 3D volumetric data along the specified axis at the specified index.

Parameters

`in`	*volume*	The 3D volumetric data to extract the slice from.
`in`	*axis*	The axis along which to extract the slice.
`in`	*index*	The index of the slice along the specified axis.

Returns

The 2D image representing the extracted slice.

Here is the call graph for this function:

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

- src/Slice.hpp
- src/Slice.cpp

4.18 Test_time Struct Reference

Collaboration diagram for Test_time:

Public Attributes

- double min
- double max
- double avg

4.18.1 Member Data Documentation

4.18.1.1 avg

```
double Test_time::avg
```

4.18.1.2 max

```
double Test_time::max
```

4.18.1.3 min

```
double Test_time::min
```

The documentation for this struct was generated from the following file:

- src/checkOutput.cpp

4.19 Volume< DATA > Class Template Reference

Class for representing volumetric data as a collection of 2D image slices.

```
#include <Volume.hpp>
```

Collaboration diagram for Volume< DATA >:

Public Member Functions

- Volume (const char ∗, const int &, const int &, const std::size_t &=4, const char ∗=".png")
  Constructor for reading thin or thick slab volumetric data from file.
- Volume (const char ∗, const std::vector< int > &, const std::size_t &=4, const char ∗=".png")
  Constructor for reading interleaved or explicitly defined ranges of volumetric data from file.
- Volume (const int &, const int &, const int &, const int &)
  Constructor for creating a user-defined volumetric data object.
- ∼Volume ()
- const int & getDepth () const
- const int & getWidth () const
- const int & getHeight () const
- const int & getChannels () const
- const std::vector< int > & getIndices () const
- void setIndices (const std::vector< int > &)
- void setVoxel (const DATA &, const int &, const int &, const int &, const int &=0)
- const DATA & getVoxel (const int &, const int &, const int &, const int &=0, const PaddingType &=PaddingType::extend, const DATA &=0) const
- void write (const char ∗, const std::size_t &=4, const char ∗=".png") const

Static Public Member Functions

- static std::vector< int > return_iota (const int &, const int &)

Protected Attributes

- int depth
- int width
- int height
- int channels
- std::vector< int > image_indices
- std::vector< Image< DATA > * > image_slices

4.19.1 Detailed Description

template<typename DATA = unsigned char>
class Volume< DATA >

Class for representing volumetric data as a collection of 2D image slices.

Template Parameters

DATA	The data type of the elements in the volumetric data.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 Volume() [1/3]

```
template<typename DATA >
Volume< DATA >::Volume (
            const char * image_filename_prefix,
            const int & index_begin,
            const int & index_end,
            const std::size_t & field_width = 4,
            const char * image_filename_suffix = ".png" )
```

Constructor for reading thin or thick slab volumetric data from file.

This constructor reads in volumetric data from a series of files in sequence along the specified axis. The files can either be thin or thick slabs, depending on the specified number of slices per image.

Parameters

in	*filename*	The base filename of the volumetric data files.
in	*axis*	The axis along which the slices are arranged in the files.
in	*slices_per_image*	The number of slices per image in the files.
in	*file_extension*	The file extension of the volumetric data files.

4.19.2.2 Volume() [2/3]

```
template<typename DATA >
Volume< DATA >::Volume (
            const char * image_filename_prefix,
            const std::vector< int > & index_vector,
            const std::size_t & field_width = 4,
            const char * image_filename_suffix = ".png" )
```

Constructor for reading interleaved or explicitly defined ranges of volumetric data from file.

This constructor reads in volumetric data from a series of files with interleaved indices, or explicitly defined ranges along the specified axis.

Parameters

in	*filename*	The base filename of the volumetric data files.
in	*ranges*	The ranges of the volumetric data files along the specified axis.
in	*file_extension*	The file extension of the volumetric data files.

4.19.2.3 Volume() [3/3]

```
template<typename DATA >
Volume< DATA >::Volume (
            const int & d,
            const int & w,
            const int & h,
            const int & c )
```

Constructor for creating a user-defined volumetric data object.

This constructor creates a user-defined volumetric data object with the specified dimensions and number of channels.

Parameters

in	*depth*	The number of slices in the volumetric data.
in	*width*	The width of each slice in the volumetric data.
in	*height*	The height of each slice in the volumetric data.
in	*channels*	The number of channels in the volumetric data.

4.19.2.4 ∼Volume()

```
template<typename DATA >
Volume< DATA >::∼Volume
```

4.19.3 Member Function Documentation

4.19.3.1 getChannels()

```
template<typename DATA >
const int & Volume< DATA >::getChannels
```

Here is the caller graph for this function:

4.19.3.2 getDepth()

```
template<typename DATA >
const int & Volume< DATA >::getDepth
```

Here is the caller graph for this function:

4.19.3.3 getHeight()

```
template<typename DATA >
const int & Volume< DATA >::getHeight
```

Here is the caller graph for this function:

4.19.3.4 getIndices()

```
template<typename DATA >
const std::vector< int > & Volume< DATA >::getIndices
```

Here is the caller graph for this function:

4.19.3.5 getVoxel()

```
template<typename DATA >
const DATA & Volume< DATA >::getVoxel (
             const int & z,
             const int & x,
             const int & y,
             const int & ch = 0,
             const PaddingType & pad = PaddingType::extend,
             const DATA & constant_pad_value = 0 ) const
```

Here is the call graph for this function: Here is the caller graph for this function:

4.19.3.6 getWidth()

```
template<typename DATA >
const int & Volume< DATA >::getWidth
```

Here is the caller graph for this function:

4.19.3.7 return_iota()

```
template<typename DATA >
std::vector< int > Volume< DATA >::return_iota (
             const int & begin,
             const int & end )  [static]
```

4.19.3.8 setIndices()

```
template<typename DATA >
void Volume< DATA >::setIndices (
             const std::vector< int > & arg_indices )
```

Here is the caller graph for this function:

4.19.3.9 setVoxel()

```
template<typename DATA >
void Volume< DATA >::setVoxel (
             const DATA & value,
             const int & z,
             const int & x,
             const int & y,
             const int & ch = 0 )
```

Here is the caller graph for this function:

4.19.3.10 write()

```
template<typename DATA >
void Volume< DATA >::write (
             const char * image_filename_prefix,
             const std::size_t & field_width = 4,
             const char * image_filename_suffix = ".png" ) const
```

4.19.4 Member Data Documentation

4.19.4.1 channels

```
template<typename DATA = unsigned char>
int Volume< DATA >::channels  [protected]
```

4.19.4.2 depth

```
template<typename DATA = unsigned char>
int Volume< DATA >::depth  [protected]
```

4.19.4.3 height

```
template<typename DATA = unsigned char>
int Volume< DATA >::height  [protected]
```

4.19.4.4 image_indices

```
template<typename DATA = unsigned char>
std::vector<int> Volume< DATA >::image_indices  [protected]
```

4.19.4.5 image_slices

```
template<typename DATA = unsigned char>
std::vector<Image<DATA> *> Volume< DATA >::image_slices  [protected]
```

4.19.4.6 width

```
template<typename DATA = unsigned char>
int Volume< DATA >::width  [protected]
```

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

- src/Volume.hpp
- src/Volume.cpp

Chapter 5

File Documentation

5.1 src/checkOutput.cpp File Reference

```
#include <iostream>
#include <vector>
#include "Image.hpp"
#include "Volume.hpp"
#include "Filter.hpp"
#include "Projection.hpp"
#include "Slice.hpp"
#include <chrono>
#include <fstream>
```

Include dependency graph for checkOutput.cpp:

Classes

- struct Test_time

Functions

- Test_time get_test_time (const std::vector< double > &times)
- void test_performance_2D ()
- void test_performance_3D ()
- void test_filter_2D ()
- int main (int argc, char **argv)

5.1.1 Function Documentation

5.1.1.1 get_test_time()

```
Test_time get_test_time (
          const std::vector< double > & times )
```

Here is the caller graph for this function:

5.1.1.2 main()

```
int main (
            int argc,
            char ** argv )
```

Here is the call graph for this function:

5.1.1.3 test_filter_2D()

```
void test_filter_2D ( )
```

Here is the call graph for this function: Here is the caller graph for this function:

5.1.1.4 test_performance_2D()

```
void test_performance_2D ( )
```

Here is the call graph for this function:

5.1.1.5 test_performance_3D()

```
void test_performance_3D ( )
```

Here is the call graph for this function:

5.2 src/Filter.cpp File Reference

```
#include "Filter.hpp"
#include "Image.hpp"
#include "Volume.hpp"
#include "Quickselect.hpp"
#include <algorithm>
#include <cmath>
#include <numeric>
#include <stdexcept>
#include <vector>
```

Include dependency graph for Filter.cpp:

5.3 src/Filter.hpp File Reference

```
#include "Image.hpp"
#include "Volume.hpp"
#include <vector>
```

Include dependency graph for Filter.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Filter2D< WORK, STORE >
- class Grayscale2D< WORK, STORE >
- class ColourBalance2D< WORK, STORE >
- class Brightness2D< WORK, STORE >
- class Conv2D< WORK, STORE >
- class Blur2D< WORK, STORE >
- class MedianBlur2D< WORK, STORE >
- class Edge2D< WORK, STORE >
- class HistogramEqualisation2D< WORK, STORE >
- class Filter3D< WORK, STORE >
- class Conv3D< WORK, STORE >
- class Blur3D< WORK, STORE >
- class MedianBlur3D< WORK, STORE >

Enumerations

- enum class BlurKernel { box , gaussian }
- enum class EdgeKernel { sobel , prewitt , scharr , robertscross }

5.3.1 Enumeration Type Documentation

5.3.1.1 BlurKernel

```
enum BlurKernel  [strong]
```

Enumerator

box	
gaussian	

5.3.1.2 EdgeKernel

```
enum EdgeKernel  [strong]
```

Enumerator

sobel	
prewitt	
scharr	
robertscross	

5.4 src/Image.cpp File Reference

```
#include "Image.hpp"
#include <algorithm>
#include <filesystem>
#include <stdexcept>
#include <type_traits>
#include <vector>
#include "stb_image.h"
#include "stb_image_write.h"
```
Include dependency graph for Image.cpp:

Macros

- #define STB_IMAGE_IMPLEMENTATION
- #define STB_IMAGE_WRITE_IMPLEMENTATION

5.4.1 Macro Definition Documentation

5.4.1.1 STB_IMAGE_IMPLEMENTATION

```
#define STB_IMAGE_IMPLEMENTATION
```

5.4.1.2 STB_IMAGE_WRITE_IMPLEMENTATION

```
#define STB_IMAGE_WRITE_IMPLEMENTATION
```

5.5 src/Image.hpp File Reference

Contains a class for representing and manipulating images.

```
#include <vector>
```
Include dependency graph for Image.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Image< DATA >

  Class for representing and manipulating images.

Enumerations

- enum class ImageAllocation { stbi_read , user_heap }

  Enumerates the types of image allocation modes supported by this library.
- enum class PaddingType { constant , extend , wrap , reflect }

  Enumerates the types of padding to use for out-of-bounds pixel accesses.

5.5.1 Detailed Description

Contains a class for representing and manipulating images.

5.5.2 Enumeration Type Documentation

5.5.2.1 ImageAllocation

```
enum ImageAllocation  [strong]
```

Enumerates the types of image allocation modes supported by this library.

Enumerator

stbi_read	
user_heap	

5.5.2.2 PaddingType

```
enum PaddingType  [strong]
```

Enumerates the types of padding to use for out-of-bounds pixel accesses.

Enumerator

constant	
extend	
wrap	
reflect	

5.6 src/main.cpp File Reference

```
#include <fstream>
#include <iostream>
```

```
#include <stdexcept>
#include <sstream>
#include <string>
#include <vector>
#include "Image.hpp"
#include "Volume.hpp"
#include "Filter.hpp"
#include "Projection.hpp"
#include "Slice.hpp"
```

Include dependency graph for main.cpp:

Functions

- std::vector< std::string > tokenise (const std::string &fullstring, const char separator)
- int main (int argc, char **argv)

5.6.1 Function Documentation

5.6.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Here is the call graph for this function:

5.6.1.2 tokenise()

```
std::vector<std::string> tokenise (
            const std::string & fullstring,
            const char separator )
```

Here is the caller graph for this function:

5.7 src/minimal.cpp File Reference

```
#include <iostream>
#include <vector>
#include "Image.hpp"
#include "Volume.hpp"
#include "Filter.hpp"
#include "Projection.hpp"
#include "Slice.hpp"
```

Include dependency graph for minimal.cpp:

Functions

- int old_main (int argc, char ∗∗argv)

5.7.1 Function Documentation

5.7.1.1 old_main()

```
int old_main (
            int argc,
            char ** argv )
```

Here is the call graph for this function:

5.8 src/Projection.cpp File Reference

```
#include "Projection.hpp"
#include "Image.hpp"
#include "Volume.hpp"
#include "Quickselect.hpp"
#include <algorithm>
#include <limits>
#include <vector>
```

Include dependency graph for Projection.cpp:

5.9 src/Projection.hpp File Reference

Contains a class for performing projections of volumetric data onto 2D images.

```
#include "Image.hpp"
#include "Volume.hpp"
#include <vector>
```

Include dependency graph for Projection.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Projection< WORK, STORE >

  Class for performing projections of volumetric data onto 2D images.

Enumerations

- enum class Intensity { min , max , mean , median }

  Enumerates the types of intensity values to use for projection.

5.9.1 Detailed Description

Contains a class for performing projections of volumetric data onto 2D images.

5.9.2 Enumeration Type Documentation

5.9.2.1 Intensity

```
enum Intensity  [strong]
```

Enumerates the types of intensity values to use for projection.

Enumerator

min	
max	
mean	
median	

5.10 src/Quickselect.cpp File Reference

```
#include "Quickselect.hpp"
#include <algorithm>
#include <vector>
```

Include dependency graph for Quickselect.cpp:

5.11 src/Quickselect.hpp File Reference

Contains a class for performing quickselect and median selection algorithms.

```
#include <vector>
```

Include dependency graph for Quickselect.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Quickselect< T >

  Class for performing quickselect and median selection algorithms on a vector.

5.11.1 Detailed Description

Contains a class for performing quickselect and median selection algorithms.

5.12 src/Slice.cpp File Reference

```
#include "Slice.hpp"
#include "Image.hpp"
#include "Volume.hpp"
#include "Projection.hpp"
#include <vector>
```

Include dependency graph for Slice.cpp:

5.13 src/Slice.hpp File Reference

Contains a class for extracting 2D slices from 3D volumetric data.

```
#include "Image.hpp"
#include "Volume.hpp"
#include "Projection.hpp"
```

Include dependency graph for Slice.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Slice< WORK, STORE >

    Class for extracting 2D slices from 3D volumetric data.

5.13.1 Detailed Description

Contains a class for extracting 2D slices from 3D volumetric data.

5.14 src/Volume.cpp File Reference

```
#include <iomanip>
#include <numeric>
#include <stdexcept>
#include <sstream>
#include <vector>
#include "Volume.hpp"
#include "Image.hpp"
```

Include dependency graph for Volume.cpp:

5.15 src/Volume.hpp File Reference

Contains a class for representing volumetric data.

```
#include "Image.hpp"
#include <vector>
```

Include dependency graph for Volume.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Volume< DATA >

  Class for representing volumetric data as a collection of 2D image slices.

Enumerations

- enum class Axis { sideview , plan , elevation }

  Enumerates the axes along which to slice or project volumetric data.

5.15.1 Detailed Description

Contains a class for representing volumetric data.

5.15.2 Enumeration Type Documentation

5.15.2.1 Axis

```
enum Axis  [strong]
```

Enumerates the axes along which to slice or project volumetric data.

Enumerator

sideview	
plan	
elevation	

Index