

Advanced Programming Group Project Fibonacci

Generated by Doxygen 1.10.0

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 Filter Class Reference	5
3.1.1 Member Function Documentation	5
3.1.1.1 addSaltPepperNoise()	5
3.1.1.2 applyBoxFilter2D()	6
3.1.1.3 applyGaussianFilter2D()	6
3.1.1.4 applyGaussianFilter3D()	6
3.1.1.5 applyMedianFilter2D()	7
3.1.1.6 applyMedianFilter3D()	7
3.1.1.7 brightness()	8
3.1.1.8 grayscale()	8
3.1.1.9 histogramEqualization()	8
3.1.1.10 threshold()	9
3.2 Image Class Reference	9
3.2.1 Member Function Documentation	10
3.2.1.1 calculateCdf()	10
3.2.1.2 clone()	11
3.2.1.3 createEmpty()	11
3.2.1.4 getFileName()	11
3.2.1.5 getHeight()	12
3.2.1.6 getPixel() [1/2]	12
3.2.1.7 getPixel() [2/2]	12
3.2.1.8 getWidth()	13
3.2.1.9 hslToRgb()	13
3.2.1.10 hsvToRgb()	14
3.2.1.11 isSameAs()	14
3.2.1.12 load()	15
3.2.1.13 operator-()	15
3.2.1.14 rgbToHsl()	16
3.2.1.15 rgbToHsv()	17
3.2.1.16 save()	17
3.2.1.17 setPixel() [1/2]	18
3.2.1.18 setPixel() [2/2]	18
3.3 Projection Class Reference	19
3.3.1 Member Function Documentation	19
3.3.1.1 AIP()	19
3.3.1.2 AIP_Median()	19

3.3.1.3 MinIP()	20
3.3.1.4 MIP()	21
3.4 Slicer Class Reference	21
3.4.1 Member Function Documentation	22
3.4.1.1 generateSlices()	22
3.4.1.2 sliceXY()	23
3.4.1.3 sliceXZ()	23
3.4.1.4 sliceYZ()	24
3.5 Volume Class Reference	24
3.5.1 Constructor & Destructor Documentation	25
3.5.1.1 Volume()	25
3.5.2 Member Function Documentation	25
3.5.2.1 createEmpty()	25
3.5.2.2 getDepth()	26
3.5.2.3 getHeight()	26
3.5.2.4 getSlice()	26
3.5.2.5 getSlices()	26
3.5.2.6 getVoxel()	27
3.5.2.7 getWidth()	27
3.5.2.8 loadVolume()	27
3.5.2.9 saveVolume()	28
3.5.2.10 setVoxel()	28
3.5.2.11 sliceXY()	28
3.5.2.12 sliceXZ()	28
3.5.2.13 sliceYZ()	29
3.5.2.14 updateSlice()	29
4 File Documentation	31
4.1 temp/Filter.cpp File Reference	31
4.1.1 Detailed Description	31
4.2 temp/Filter.h File Reference	31
4.2.1 Detailed Description	32
4.3 Filter.h	32
4.4 temp/Image.cpp File Reference	33
4.4.1 Detailed Description	33
4.5 temp/Image.h File Reference	34
4.5.1 Detailed Description	34
4.6 Image.h	34
4.7 temp/main.cpp File Reference	35
4.7.1 Detailed Description	35
4.8 temp/Projection.cpp File Reference	35
4.8.1 Detailed Description	36

4.9 temp/Projection.h File Reference	36
4.9.1 Detailed Description	36
4.10 Projection.h	36
4.11 temp/Slice.cpp File Reference	36
4.11.1 Detailed Description	37
4.12 temp/Slice.h File Reference	37
4.12.1 Detailed Description	37
4.13 Slice.h	37
4.14 temp/Utils.cpp File Reference	38
4.14.1 Detailed Description	38
4.14.2 Function Documentation	38
4.14.2.1 apply1DConvolution2D()	38
4.14.2.2 apply1DConvolution3D()	39
4.14.2.3 convertNumberToString()	39
4.14.2.4 createOutputDirectory()	40
4.14.2.5 extractFilename()	40
4.14.2.6 extractFilenameWithoutExtension()	40
4.14.2.7 generate1DGaussianKernel()	40
4.14.2.8 merge()	41
4.14.2.9 mergeSort()	41
4.14.2.10 partition()	42
4.14.2.11 quickSelect()	42
4.15 temp/Utils.h File Reference	43
4.15.1 Detailed Description	43
4.15.2 Function Documentation	43
4.15.2.1 apply1DConvolution2D()	43
4.15.2.2 apply1DConvolution3D()	44
4.15.2.3 convertNumberToString()	44
4.15.2.4 createOutputDirectory()	45
4.15.2.5 extractFilename()	45
4.15.2.6 extractFilenameWithoutExtension()	45
4.15.2.7 generate1DGaussianKernel()	45
4.15.2.8 merge()	46
4.15.2.9 mergeSort()	46
4.15.2.10 partition()	47
4.15.2.11 quickSelect()	47
4.16 Utils.h	48
4.17 temp/Volume.cpp File Reference	48
4.17.1 Detailed Description	48
4.18 temp/Volume.h File Reference	48
4.18.1 Detailed Description	49
4.19 Volume.h	49

Chapter 1

Class Index

1.1 Class List

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

Filter	5
Image	9
Projection	19
Slicer	21
Volume	24

Chapter 2

File Index

2.1 File List

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

temp/ Filter.cpp	31
temp/ Filter.h	31
temp/ Image.cpp	33
temp/ Image.h	34
temp/ main.cpp	35
temp/ Projection.cpp	35
temp/ Projection.h	36
temp/ Slice.cpp	36
temp/ Slice.h	37
temp/ utils.cpp	38
temp/ utils.h	43
temp/ Volume.cpp	48
temp/ Volume.h	48

Chapter 3

Class Documentation

3.1 Filter Class Reference

Static Public Member Functions

- static void [grayscale](#) ([Image](#) &image)
- static void [brightness](#) ([Image](#) &image, int brightness, bool autoBrightness)
- static void [histogramEqualization](#) ([Image](#) &image, bool hsv)
- static void [threshold](#) ([Image](#) &image, int threshold, bool useHsv)
- static void [addSaltPepperNoise](#) ([Image](#) &image, float noisePercentage)
- static void [applyMedianFilter2D](#) ([Image](#) &image, int kernel_size)
- static void [applyBoxFilter2D](#) ([Image](#) &image, int kernel_size)
- static void [applyGaussianFilter2D](#) ([Image](#) &image, int kernel_size, double sigma=2.0)
- static void **sobel** ([Image](#) &image)
- static void **robert** ([Image](#) &image)
- static void **scharr** ([Image](#) &image)
- static void **prewitt** ([Image](#) &image)
- static void [applyGaussianFilter3D](#) ([Volume](#) &volume, int kernelSize, double sigma)
Apply a Guassian filter to a 3d volume.
- static void [applyMedianFilter3D](#) ([Volume](#) &volume, int kernelSize)
Apply a median filter to a 3d volume.

3.1.1 Member Function Documentation

3.1.1.1 addSaltPepperNoise()

```
void Filter::addSaltPepperNoise (
    Image & image,
    float noisePercentage ) [static]
```

Adds salt-and-pepper noise to the given image.

Parameters

<i>image</i>	The image to add noise to.
<i>noisePercentage</i>	The percentage of pixels to noise.

The salt-and-pepper noise filter randomly sets a percentage of pixels to either 0 (black) or 255 (white). The noise percentage determines the number of pixels to noise in the image. The filter is applied to the grayscale image or all channels in the RGB image.

3.1.1.2 applyBoxFilter2D()

```
void Filter::applyBoxFilter2D (
    Image & image,
    int kernel_size ) [static]
```

Applies a box blur filter to the given image.

Parameters

<i>image</i>	The image to apply the filter to.
<i>kernel_size</i>	The size of the kernel used for blurring.

Applies a simple box blur filter to the image using the specified kernel size. The box blur filter is a simple averaging of the surrounding pixels. The kernel size determines the number of surrounding pixels to consider.

3.1.1.3 applyGaussianFilter2D()

```
void Filter::applyGaussianFilter2D (
    Image & image,
    int kernel_size,
    double sigma = 2.0 ) [static]
```

Applies a Gaussian blur filter to the given image.

Parameters

<i>image</i>	The image to apply the filter to.
<i>kernel_size</i>	The size of the kernel used for blurring.
<i>sigma</i>	The standard deviation of the Gaussian distribution.

Generates a Gaussian kernel using the specified kernel size. Applies the Gaussian kernel to the image to perform the blur.

3.1.1.4 applyGaussianFilter3D()

```
void Filter::applyGaussianFilter3D (
    Volume & volume,
    int kernelSize,
    double sigma ) [static]
```

Apply a Gaussian filter to a 3d volume.

Parameters

<i>volume</i>	Reference to the Volume object to be filtered. The volume is updated in-place with the filtered results.
<i>kernelSize</i>	The size of the Gaussian kernel to be used for filtering, which should be a positive odd integer.
<i>sigma</i>	The standard deviation of the Gaussian kernel.

This function applies a Gaussian filter to the given 3D volume using a separable approach. It first generates a 1D Gaussian kernel based on the specified kernel size and sigma (standard deviation). The filtering is performed in three steps, applying the 1D convolution along the X, Y, and Z axes sequentially. This separable approach is more efficient than a direct 3D convolution, especially for larger kernel sizes. The original volume is updated with the filtered results.

3.1.1.5 applyMedianFilter2D()

```
void Filter::applyMedianFilter2D (
    Image & image,
    int kernel_size ) [static]
```

Applies the image median blur filter to the given image.

Parameters

<i>image</i>	The image to apply the filter to.
<i>kernel_size</i>	The size of the kernel used for blurring.

The image median blur filter calculates the median value of the RGB channels for each pixel and assigns the median value to the pixel. The filter is applied to the image by looping through each pixel and its surrounding pixels within the specified kernel size. The median value is calculated for the red, green, and blue channels separately, and then assigned to the current pixel. Source reference: <https://chat.openai.com/share/bfc5caa6-324f-4f8a-9967-d4a960433151>

3.1.1.6 applyMedianFilter3D()

```
void Filter::applyMedianFilter3D (
    Volume & volume,
    int kernelSize ) [static]
```

Apply a median filter to a 3d volume.

Parameters

<i>volume</i>	Reference to the Volume object to be filtered. The volume is updated in-place with the filtered results.
<i>kernelSize</i>	The size of the cubic neighborhood around each voxel considered for median filtering, which should be a positive odd integer to ensure a single median value can be computed.

This function filters the given 3D volume using a median filter. The function operates by moving through each voxel in the volume, considering a cubic neighborhood around it defined by the kernel size. It calculates the median value within this neighborhood and sets the voxel's new value to this median.

3.1.1.7 brightness()

```
void Filter::brightness (
    Image & image,
    int brightness = 0,
    bool autoBrightness = false ) [static]
```

Applies a brightness filter to the given image.

Parameters

<i>image</i>	The image to apply the filter to.
<i>brightness</i>	The brightness value to apply to the image.
<i>autoBrightness</i>	Flag to enable automatic brightness calculation.

The brightness filter adjusts the brightness of the image by adding a constant value. The brightness value can be manually set or automatically calculated based on the image. The autoBrightness flag enables automatic brightness calculation based on the average pixel value.

3.1.1.8 grayscale()

```
void Filter::grayscale (
    Image & image ) [static]
```

Applies a grayscale filter to the given image.

Parameters

<i>image</i>	The image to apply the filter to.
--------------	-----------------------------------

The grayscale filter converts the RGB image to a single-channel grayscale image. The grayscale value is calculated using the formula: $\text{gray} = 0.2126 * r + 0.7152 * g + 0.0722 * b$

3.1.1.9 histogramEqualization()

```
void Filter::histogramEqualization (
    Image & image,
    bool hsv ) [static]
```

Applies a histogram equalization filter to the given image.

Parameters

<i>image</i>	The image to apply the filter to.
<i>hsv</i>	Flag to indicate whether to use HSV or HSL color space.

The histogram equalization filter enhances the contrast of the image by redistributing the pixel intensity values. The filter is applied to the grayscale image or each channel of the RGB image separately.

3.1.1.10 threshold()

```
void Filter::threshold (
    Image & image,
    int threshold,
    bool useHsv = false ) [static]
```

Applies a threshold filter to the given image.

Parameters

<i>image</i>	The image to apply the filter to.
<i>threshold</i>	The threshold value to use for binarization.
<i>useHsv</i>	Flag to enable thresholding in HSV color space.

The threshold filter converts the image to a binary image based on the threshold value. The filter is applied to the grayscale image or the V channel in the HSV color space. Pixels with intensity values below the threshold are set to 0 (black), and above to 255 (white).

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

- temp/[Filter.h](#)
- temp/[Filter.cpp](#)

3.2 Image Class Reference

Public Member Functions

- int [getWidth](#) () const
Get the Width object.
- int [getHeight](#) () const
Get the Height object.
- void [setPixel](#) (int x, int y, unsigned char value)
Set the Pixel object.
- unsigned char [getPixel](#) (int x, int y) const
Get the Pixel object.
- [Image clone](#) () const
clone the image
- [Image operator-](#) (const [Image](#) &other) const
 - *operator overloading*
- std::string [getFileName](#) () const
Get the File Name object.
- void [save](#) (const std::string &path) const
save the image
- void [createEmpty](#) (int width, int height, int channels)
Create a Empty object.
- void [load](#) (const std::string &path)
load the image
- bool [isSameAs](#) (const [Image](#) &other) const
check if the image is same as the other image
- **Image** (int width=0, int height=0, int channels=3)
- unsigned char [getPixel](#) (int x, int y, int ch) const
Get the Pixel object.
- void [setPixel](#) (int x, int y, int ch, unsigned char value)
Set the Pixel object.

Static Public Member Functions

- static void [rgbToHsv](#) (float r, float g, float b, float &h, float &s, float &v)
rgb to hsv conversion
- static void [hsvToRgb](#) (float h, float s, float v, int &r, int &g, int &b)
hsv to rgb conversion
- static void [rgbToHsl](#) (float r, float g, float b, float &h, float &s, float &l)
rgb to hsl conversion
- static void [hslToRgb](#) (float h, float s, float l, int &r, int &g, int &b)
hsl to rgb conversion
- static void [calculateCdf](#) (const std::vector< int > &histogram, std::vector< int > &cdf)
calculate cdf

Public Attributes

- unsigned char * **data**
constructor
- int **w**
- int **h**
- int **c**

3.2.1 Member Function Documentation

3.2.1.1 calculateCdf()

```
void Image::calculateCdf (
    const std::vector< int > & histogram,
    std::vector< int > & cdf ) [static]
```

calculate cdf

Parameters

<i>histogram</i>	input histogram
<i>cdf</i>	output cdf

Process:

- Calculate the cumulative distribution function (CDF) from the histogram.
- The CDF is calculated by summing the histogram values.
- The CDF values are stored in the cdf vector.
- The histogram values are in the range [0, 255].
- The cdf values are in the range [0, total number of pixels].
- The cdf vector is resized to match the histogram size.

3.2.1.2 clone()

```
Image Image::clone ( ) const
```

clone the image

Returns

[Image](#) reference to the cloned image

Process:

- Clone the image.
- Return the reference to the cloned image.

3.2.1.3 createEmpty()

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

Create a Empty object.

Parameters

<i>width</i>	input width
<i>height</i>	input height
<i>channels</i>	input channels

Process:

- Create an empty image with the specified width, height, and number of channels.

3.2.1.4 getFileName()

```
std::string Image::getFileName ( ) const
```

Get the File Name object.

Returns

std::string name of the file

Process:

- Get the name of the file.
- Return the name of the file.

3.2.1.5 getHeight()

```
int Image::getHeight ( ) const
```

Get the Height object.

Returns

int height of the image

Process:

- Get the height of the image.

3.2.1.6 getPixel() [1/2]

```
unsigned char Image::getPixel (
    int x,
    int y ) const
```

Get the Pixel object.

Parameters

x	input x coordinate
y	input y coordinate

Returns

unsigned char

Process:

- Get the pixel value at the given coordinates.
- If the coordinates are out of bounds, the function returns 0.

3.2.1.7 getPixel() [2/2]

```
unsigned char Image::getPixel (
    int x,
    int y,
    int ch ) const
```

Get the Pixel object.

Parameters

<i>x</i>	input x coordinate
<i>y</i>	input y coordinate
<i>ch</i>	input channel

Returns

unsigned char pixel value

Process:

- Get the pixel value at the given coordinates and channel.
- If the coordinates or channel are out of bounds, the function throws an exception.
- Return the pixel value.

3.2.1.8 getWidth()

```
int Image::getWidth ( ) const
```

Get the Width object.

Returns

int width of the image

Process:

- Get the width of the image.

3.2.1.9 hslToRgb()

```
void Image::hslToRgb (
    float h,
    float s,
    float l,
    int & r,
    int & g,
    int & b ) [static]
```

hsl to rgb conversion

Parameters

<i>h</i>	input hue value
<i>s</i>	input saturation value
<i>l</i>	input lightness value
<i>r</i>	output red value
<i>g</i>	output green value
<i>b</i>	output blue value

Process:

- Convert the HSL color to RGB color.
- Return the RGB color.
- The HSL color values are in the range [0, 360] for hue and [0, 1] for saturation and lightness.
- The RGB color values are in the range [0, 255].

3.2.1.10 hsvToRgb()

```
void Image::hsvToRgb (
    float h,
    float s,
    float v,
    int & r,
    int & g,
    int & b ) [static]
```

hsv to rgb conversion

Parameters

<i>h</i>	input hue value
<i>s</i>	input saturation value
<i>v</i>	input value
<i>r</i>	output red value
<i>g</i>	output green value
<i>b</i>	output blue value

Process:

- Convert the HSV color to RGB color.
- Return the RGB color.
- The HSV color values are in the range [0, 360] for hue and [0, 1] for saturation and value.
- The RGB color values are in the range [0, 255].
- The hue value is in the range [0, 360].

3.2.1.11 isSameAs()

```
bool Image::isSameAs (
    const Image & other ) const
```

check if the image is same as the other image

Parameters

<i>other</i>	input image
--------------	-------------

Returns

true if the image is same
false if the image is not same

Process:

- Check if the image is the same as the other image.
- Return true if the images are the same.
- Return false if the images are not the same.

3.2.1.12 load()

```
void Image::load (  
    const std::string & path )
```

load the image

Parameters

<i>path</i>	input path
-------------	------------

Process:

- Load the image from the specified path.
- If the image is not loaded successfully, an exception is thrown.

3.2.1.13 operator-()

```
Image Image::operator- (  
    const Image & other ) const
```

- operator overloading

Parameters

<i>other</i>	input image
--------------	-------------

Returns

[Image](#) resultant image

Process:

- Subtract the pixel values of the two images.
- Return the resultant image.
- If the images have different sizes, the resultant image will have the size of the larger image.
- If the images have different number of channels, the resultant image will have the number of channels of the larger image.
- If the pixel value is negative, it is set to 0.

3.2.1.14 rgbToHsl()

```
void Image::rgbToHsl (
    float r,
    float g,
    float b,
    float & h,
    float & s,
    float & l ) [static]
```

rgb to hsl conversion

Parameters

<i>r</i>	input red value
<i>g</i>	input green value
<i>b</i>	input blue value
<i>h</i>	output hue value
<i>s</i>	output saturation value
<i>l</i>	output lightness value

Process:

- Convert the RGB color to HSL color.
- Return the HSL color.
- The RGB color values are in the range [0, 255].
- The HSL color values are in the range [0, 360] for hue and [0, 1] for saturation and lightness.
- The hue value is in the range [0, 360].

3.2.1.15 rgbToHsv()

```
void Image::rgbToHsv (
    float r,
    float g,
    float b,
    float & h,
    float & s,
    float & v ) [static]
```

rgb to hsv conversion

Parameters

<i>r</i>	input red value
<i>g</i>	input green value
<i>b</i>	input blue value
<i>h</i>	output hue value
<i>s</i>	output saturation value
<i>v</i>	output value

Process:

- Convert the RGB color to HSV color.
- Return the HSV color.
- The RGB color values are in the range [0, 255].
- The HSV color values are in the range [0, 360] for hue and [0, 1] for saturation and value.
- The hue value is in the range [0, 360].

3.2.1.16 save()

```
void Image::save (
    const std::string & path ) const
```

save the image

Parameters

<i>path</i>	input path
-------------	------------

Process:

- Save the image to the specified path.
- If the image is not saved successfully, an exception is thrown.

3.2.1.17 `setPixel()` [1/2]

```
void Image::setPixel (
    int x,
    int y,
    int ch,
    unsigned char value )
```

Set the Pixel object.

Parameters

<i>x</i>	input x coordinate
<i>y</i>	input y coordinate
<i>ch</i>	input channel
<i>value</i>	input value

Process:

- Set the pixel value at the given coordinates and channel.

If the coordinates or channel are out of bounds, the function throws an exception.

- Set the pixel value.

3.2.1.18 `setPixel()` [2/2]

```
void Image::setPixel (
    int x,
    int y,
    unsigned char value )
```

Set the Pixel object.

Parameters

<i>x</i>	input x coordinate
<i>y</i>	input y coordinate
<i>value</i>	input value

Process:

- Set the pixel value at the given coordinates.
- If the coordinates are out of bounds, the function does nothing.

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

- [temp/Image.h](#)
- [temp/Image.cpp](#)

3.3 Projection Class Reference

Static Public Member Functions

- static [Image MIP](#) (const [Volume](#) &volume, int minIndex, int maxIndex)
Apply max intensity projection to 3d volume.
- static [Image MinIP](#) (const [Volume](#) &volume, int minIndex, int maxIndex)
Apply min intensity projection to 3d volume.
- static [Image AIP](#) (const [Volume](#) &volume, int minIndex, int maxIndex)
Apply average intensity projection to 3d volume.
- static [Image AIP_Median](#) (const [Volume](#) &volume, int minIndex, int maxIndex)
Apply median intensity projection to 3d volume.

3.3.1 Member Function Documentation

3.3.1.1 AIP()

```
Image Projection::AIP (
    const Volume & volume,
    int minIndex = 1,
    int maxIndex = -1 ) [static]
```

Apply average intensity projection to 3d volume.

Parameters

<i>volume</i>	input volume
<i>minIndex</i>	input min index
<i>maxIndex</i>	input max index

Returns

[Image](#) resultant image

Process:

- For each pixel in the output image, find the average intensity value along the z-axis.
- The output image will have the same width and height as the input volume.
- The intensity value at each pixel is the average intensity value found along the z-axis. Reference: <https://chat.openai.com/share/53c71398-4690-4906-a012-da51c683b7e1>

3.3.1.2 AIP_Median()

```
Image Projection::AIP_Median (
    const Volume & volume,
    int minIndex = 1,
    int maxIndex = -1 ) [static]
```

Apply median intensity projection to 3d volume.

Parameters

<i>volume</i>	input volume
<i>minIndex</i>	input min index
<i>maxIndex</i>	input max index

Returns

[Image](#) resultant image

Process:

- For each pixel in the output image, find the median intensity value along the z-axis.
- The output image will have the same width and height as the input volume.
- The intensity value at each pixel is the median intensity value found along the z-axis.
- The median intensity value is calculated using the quick select algorithm.

Reference: <https://chat.openai.com/share/53c71398-4690-4906-a012-da51c683b7e1>

3.3.1.3 MinIP()

```
Image Projection::MinIP (  
    const Volume & volume,  
    int minIndex = 1,  
    int maxIndex = -1 ) [static]
```

Apply min intensity projection to 3d volume.

Parameters

<i>volume</i>	input volume
<i>minIndex</i>	input min index
<i>maxIndex</i>	input max index

Returns

[Image](#) resultant image

Process:

- For each pixel in the output image, find the minimum intensity value along the z-axis.
- The output image will have the same width and height as the input volume.
- The intensity value at each pixel is the minimum intensity value found along the z-axis. Reference: <https://chat.openai.com/share/53c71398-4690-4906-a012-da51c683b7e1>

3.3.1.4 MIP()

```
Image Projection::MIP (
    const Volume & volume,
    int minIndex = 1,
    int maxIndex = -1 ) [static]
```

Apply max intensity projection to 3d volume.

Parameters

<i>volume</i>	input volume
<i>minIndex</i>	input min index
<i>maxIndex</i>	input max index

Returns

Image resultant image

Process:

- For each pixel in the output image, find the maximum intensity value along the z-axis.
- The output image will have the same width and height as the input volume.
- The intensity value at each pixel is the maximum intensity value found along the z-axis. Reference: <https://chat.openai.com/share/53c71398-4690-4906-a012-da51c683b7e1>

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

- temp/[Projection.h](#)
- temp/[Projection.cpp](#)

3.4 Slicer Class Reference

Static Public Member Functions

- static **Image sliceXY** (const **Volume** &volume, int z)
Slice the volume in XY plane at z.
- static **Image sliceYZ** (const **Volume** &volume, int x)
Slice the volume in YZ plane at x.
- static **Image sliceXZ** (const **Volume** &volume, int y)
Slice the volume in XZ plane at y.
- static void **generateSlices** (const **Volume** &volume, const std::string &sliceFolder, int xCoordinate, int yCoordinate)
Generate slices at specific coordinates.

3.4.1 Member Function Documentation

3.4.1.1 generateSlices()

```
void Slicer::generateSlices (
    const Volume & volume,
    const std::string & sliceFolder,
    int xCoordinate,
    int yCoordinate ) [static]
```

Generate slices at specific coordinates.

Parameters

<i>volume</i>	input volume
<i>sliceFolder</i>	input slice folder
<i>xCoordinate</i>	input x coordinate
<i>yCoordinate</i>	input y coordinate

Process:

- Ensure the slice folder exists.
- Generate and save YZ slice at xCoordinate.
- Generate and save XZ slice at yCoordinate.

3.4.1.2 sliceXY()

```
Image Slicer::sliceXY (  
    const Volume & volume,  
    int z ) [static]
```

Slice the volume in XY plane at z.

Parameters

<i>volume</i>	input volume
<i>z</i>	input z coordinate

Returns

[Image](#) resultant image

Process:

- Generate a slice in the XY plane at a given z-depth.
- If the z-coordinate is out of range, print an error message and return an empty image.

3.4.1.3 sliceXZ()

```
Image Slicer::sliceXZ (  
    const Volume & volume,  
    int y ) [static]
```

Slice the volume in XZ plane at y.

Parameters

<i>volume</i>	input volume
<i>y</i>	input y coordinate

Returns

[Image](#) resultant image

Process:

- Generate a slice in the XZ plane at a given y-coordinate.
- If the y-coordinate is out of range, print an error message and return an empty image.

3.4.1.4 sliceYZ()

```
Image Slicer::sliceYZ (
    const Volume & volume,
    int x ) [static]
```

Slice the volume in YZ plane at x.

Parameters

<i>volume</i>	input volume
<i>x</i>	input x coordinate

Returns

[Image](#) resultant image

- Process:
- Generate a slice in the YZ plane at a given x-coordinate.
 - If the x-coordinate is out of range, print an error message and return an empty image.

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

- [temp/Slice.h](#)
- [temp/Slice.cpp](#)

3.5 Volume Class Reference

Public Member Functions

- **Volume ()**
Construct a new [Volume](#) object.
- **Volume (const std::string &directoryPath)**
Construct a new [Volume](#) object from the directory path.
- **Image getSlice (int index) const**
Get the [Slice](#) object.
- void **loadVolume (const std::string &directoryPath)**
load the volume from the directory path
- void **createEmpty (int width, int height, int depth)**

- Create a Empty object.*

 - int `getWidth ()` const

Get the Width object.
- int `getHeight ()` const

Get the Height object.
- int `getDepth ()` const

Get the Depth object.
- unsigned char `getVoxel (int x, int y, int z)` const

Get the Voxel object.
- void `setVoxel (unsigned char v, int x, int y, int z)`

Set the Voxel object.
- void `updateSlice (int z, Image newData)`

Update the Slice object.
- void `saveVolume (const std::string &outputDirectory)` const

save the volume to the output directory
- `Image sliceXY (int zCoordinate)` const

slice the volume in the XY plane
- `Image sliceXZ (int yCoordinate)` const

slice the volume in the XZ plane
- `Image sliceYZ (int xCoordinate)` const

slice the volume in the YZ plane
- const std::vector< `Image` > & `getSlices ()` const

Get the Slices object.

3.5.1 Constructor & Destructor Documentation

3.5.1.1 Volume()

```
Volume::Volume (
    const std::string & directoryPath )
```

Construct a new `Volume` object from the directory path.

Parameters

<code>directoryPath</code>	input directory path
----------------------------	----------------------

3.5.2 Member Function Documentation

3.5.2.1 createEmpty()

```
void Volume::createEmpty (
    int width,
    int height,
    int depth )
```

Create a Empty object.

Parameters

<i>width</i>	input width
<i>height</i>	input height
<i>depth</i>	input depth

3.5.2.2 getDepth()

```
int Volume::getDepth ( ) const
```

Get the Depth object.

Returns

int resultant depth

3.5.2.3 getHeight()

```
int Volume::getHeight ( ) const
```

Get the Height object.

Returns

int resultant height

3.5.2.4 getSlice()

```
Image Volume::getSlice (
    int index ) const
```

Get the Slice object.

Parameters

<i>index</i>	input index
--------------	-------------

Returns

[Image](#) resultant image

3.5.2.5 getSlices()

```
const std::vector< Image > & Volume::getSlices ( ) const [inline]
```

Get the Slices object.

Returns

const std::vector<Image>& resultant vector of images

3.5.2.6 getVoxel()

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

Get the Voxel object.

Parameters

x	input x
y	input y
z	input z

Returns

unsigned char resultant unsigned char

3.5.2.7 getWidth()

```
int Volume::getWidth ( ) const
```

Get the Width object.

Returns

int resultant width

3.5.2.8 loadVolume()

```
void Volume::loadVolume (
    const std::string & directoryPath )
```

load the volume from the directory path

Parameters

<i>directoryPath</i>	input directory path
----------------------	----------------------

3.5.2.9 saveVolume()

```
void Volume::saveVolume (
    const std::string & outputDirectory ) const
```

save the volume to the output directory

Parameters

<i>outputDirectory</i>	input output directory
------------------------	------------------------

3.5.2.10 setVoxel()

```
void Volume::setVoxel (
    unsigned char v,
    int x,
    int y,
    int z )
```

Set the Voxel object.

Parameters

<i>v</i>	input v
<i>x</i>	input x
<i>y</i>	input y
<i>z</i>	input z

3.5.2.11 sliceXY()

```
Image Volume::sliceXY (
    int zCoordinate ) const
```

slice the volume in the XY plane

Parameters

<i>zCoordinate</i>	input z coordinate
--------------------	--------------------

Returns

[Image](#) resultant image

3.5.2.12 sliceXZ()

```
Image Volume::sliceXZ (
    int yCoordinate ) const
```

slice the volume in the XZ plane

Parameters

<i>yCoordinate</i>	input y coordinate
--------------------	--------------------

Returns

[Image](#) resultant image

3.5.2.13 sliceYZ()

```
Image Volume::sliceYZ (
    int xCoordinate ) const
```

slice the volume in the YZ plane

Parameters

<i>xCoordinate</i>	input x coordinate
--------------------	--------------------

Returns

[Image](#) resultant image

3.5.2.14 updateSlice()

```
void Volume::updateSlice (
    int z,
    Image newData )
```

Update the Slice object.

Parameters

<i>z</i>	input z
<i>newData</i>	input newData

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

- temp/[Volume.h](#)
- temp/[Volume.cpp](#)

Chapter 4

File Documentation

4.1 temp/Filter.cpp File Reference

```
#include <iostream>
#include <vector>
#include <cassert>
#include <algorithm>
#include <cmath>
#include <cstdlib>
#include <tuple>
#include <ctime>
#include "utils.h"
#include "Filter.h"
#include "Image.h"
#include <chrono>
```

4.1.1 Detailed Description

Group Name: Fibonacci Name: Yibin Gao Github_name: edsml-kg23 Name: Ruihan He Github_name: edsml-rh323 Name: Yu Yin Github_name: acse-yy923 Name: Sara Lakatos Github_name: acse-sl4623 Name: Manawi Kahie Github_name: acse-mk1923 Name: Wenyi Yang Github_name: acse-wy1023

4.2 temp/Filter.h File Reference

```
#include "Image.h"
#include "Volume.h"
#include <string>
#include <vector>
```

Classes

- class [Filter](#)

Enumerations

- enum class **Axis** { **X**, **Y**, **Z** }

4.2.1 Detailed Description

Group Name: Fibonacci Name: Yibin Gao Github_name: edsml-kg23 Name: Ruihan He Github_name: edsml-rh323 Name: Yu Yin Github_name: acse-yy923 Name: Sara Lakatos Github_name: acse-sl4623 Name: Manawi Kahie Github_name: acse-mk1923 Name: Wenyi Yang Github_name: acse-wy1023

4.3 Filter.h

[Go to the documentation of this file.](#)

```
00001
00012 #ifndef FILTER_H
00013 #define FILTER_H
00014
00015 #include "Image.h"
00016 #include "Volume.h"
00017 #include <string>
00018 #include <vector>
00019
00020
00021 enum class Axis { X, Y, Z };
00022
00023
00024 class Filter {
00025 private:
00026     enum class PaddingMethod {
00027         Zero, // Zero padding
00028         Reflect, // Reflect padding
00029         Copy // Copy edge padding
00030     };
00031
00032     /*
00033     * Applies a generic edge detection filter to an image using specified gradient operators.
00034     *
00035     * @param image Reference to an Image object containing the source image data.
00036     *               The function directly modifies the image object to store the result.
00037     * @param gx     The horizontal gradient operator (kernel) for edge detection.
00038     * @param gy     The vertical gradient operator (kernel) for edge detection.
00039     * @param padding Enum value specifying the padding method to use (Zero, Reflect, Copy).
00040     *
00041     * Process:
00042     * - Computes horizontal and vertical gradients at each pixel using gx and gy kernels.
00043     * - Calculates the gradient magnitude as the Euclidean norm of the horizontal and vertical gradients.
00044     * - Normalizes the gradient magnitude to fit within the range of 0 to 255.
00045     * - Supports different padding methods to handle edge pixels.
00046     * - reference: https://chat.openai.com/share/6133745c-f97e-465e-a3a8-a39cfe72eeaa
00047     */
00048     static void applyEdgeDetection(Image &image, const int gx[3][3], const int gy[3][3],
00049                                     PaddingMethod padding = PaddingMethod::Reflect);
00050
00051 public:
00052
00062     static void grayscale(Image &image);
00063
00075     static void brightness(Image &image, int brightness, bool autoBrightness);
00076
00086     static void histogramEqualization(Image &image, bool hsv);
00087
00099     static void threshold(Image &image, int threshold, bool useHsv);
00100
00111     static void addSaltPepperNoise(Image &image, float noisePercentage);
00112
00126     static void applyMedianFilter2D(Image& image, int kernel_size);
00127
00139     static void applyBoxFilter2D(Image& image, int kernel_size);
00140
00151     static void applyGaussianFilter2D(Image& image, int kernel_size, double sigma = 2.0);
00152
00153     /*
00154     * Applies the Sobel edge detection filter to an image.
00155     */
00156 }
```

```

00156 * @param image Reference to an Image object that will be processed.
00157 *         The result is stored directly in this object.
00158 *
00159 * Process:
00160 * - Utilizes predefined Sobel operators for horizontal and vertical gradient calculation.
00161 * - Invokes applyEdgeDetection with the Sobel operators to detect edges in the image.
00162 * - reference: https://en.wikipedia.org/wiki/Sobel\_operator
00163 */
00164 static void sobel(Image &image);
00165
00166 /*
00167 * Applies the Robert Cross edge detection filter to an image.
00168 *
00169 * @param image Reference to an Image object that will be processed.
00170 *         The result is stored directly in this object.
00171 *
00172 * Process:
00173 * - Uses predefined Robert Cross operators for horizontal and vertical gradient calculation.
00174 * - Invokes applyEdgeDetection with the Robert Cross operators to perform the edge detection.
00175 * - reference: https://en.wikipedia.org/wiki/Roberts\_Cross
00176 */
00177 static void robert(Image &image);
00178
00179 /*
00180 * Applies the Scharr edge detection filter to an image.
00181 *
00182 * @param image Reference to an Image object that will be processed.
00183 *         The result is stored directly in this object.
00184 *
00185 * Process:
00186 * - Uses predefined Scharr operators for more accurate horizontal and vertical gradient calculation.
00187 * - Invokes applyEdgeDetection with the Scharr operators to perform the edge detection.
00188 * - reference: https://theailearner.com/tag/scharr-operator/
00189 */
00190 static void scharr(Image &image);
00191
00192 /*
00193 * Applies the Prewitt edge detection filter to an image.
00194 *
00195 * @param image Reference to an Image object that will be processed.
00196 *         The result is stored directly in this object.
00197 *
00198 * Process:
00199 * - Uses predefined Prewitt operators for horizontal and vertical gradient calculation.
00200 * - Invokes applyEdgeDetection with the Prewitt operators to perform the edge detection.
00201 * - reference: https://en.wikipedia.org/wiki/Prewitt\_operator
00202 */
00203 static void prewitt(Image &image);
00204
00220 static void applyGaussianFilter3D(Volume& volume, int kernelSize, double sigma);
00221
00234 static void applyMedianFilter3D(Volume& volume, int kernelSize);
00235
00236 };
00237 #endif
00238

```

4.4 temp/Image.cpp File Reference

```

#include "Image.h"
#include "stb_image.h"
#include "stb_image_write.h"
#include <iostream>
#include <cmath>
#include <algorithm>
#include <cstring>

```

4.4.1 Detailed Description

Group Name: Fibonacci Name: Yibin Gao Github_name: edsml-kg23 Name: Ruihan He Github_name: edsml-rh323 Name: Yu Yin Github_name: acse-yy923 Name: Sara Lakatos Github_name: acse-sl4623 Name: Manawi Kahie Github_name: acse-mk1923 Name: Wenyi Yang Github_name: acse-wy1023

4.5 temp/Image.h File Reference

```
#include <string>
#include <vector>
```

Classes

- class [Image](#)

4.5.1 Detailed Description

Group Name: Fibonacci Name: Yibin Gao Github_name: edsml-kg23 Name: Ruihan He Github_name: edsml-rh323 Name: Yu Yin Github_name: acse-yy923 Name: Sara Lakatos Github_name: acse-sl4623 Name: Manawi Kahie Github_name: acse-mk1923 Name: Wenyi Yang Github_name: acse-wy1023

4.6 Image.h

[Go to the documentation of this file.](#)

```
00001
00013 // Image.h
00014 #ifndef IMAGE_H
00015 #define IMAGE_H
00016
00017 #include <string>
00018 #include <vector>
00019
00020 class Image {
00021 public:
00022     //3d slicing
00023
00032     int getWidth() const;
00033
00042     int getHeight() const;
00043
00055     void setPixel(int x, int y, unsigned char value);
00056
00068     unsigned char getPixel(int x, int y) const;
00069
00079     Image clone() const;
00080
00094     Image operator-(const Image& other) const;
00095
00105     std::string getFileName() const;
00106
00116     void save(const std::string& path) const;
00117
00128     void createEmpty(int width, int height, int channels);
00129
00139     void load(const std::string& path);
00140
00153     bool isSameAs(const Image& other) const;
00154
00159     unsigned char* data;
00160     int w, h, c;
00161
00162     // Constructor
00163     Image(int width = 0, int height = 0, int channels = 3);
00164
00165     // getter functions
00179     unsigned char getPixel(int x, int y, int ch) const;
00180
00194     void setPixel(int x, int y, int ch, unsigned char value);
00195
00196     // Utility functions declarations
00197
00215     static void rgbToHsv(float r, float g, float b, float &h, float &s, float &v);
00216
```



```

00234     static void hsvToRgb(float h, float s, float v, int &r, int &g, int &b);
00235
00254     static void rgbToHsl(float r, float g, float b, float &h, float &s, float &l);
00255
00272     static void hslToRgb(float h, float s, float l, int &r, int &g, int &b);
00273
00288     static void calculateCdf(const std::vector<int>& histogram, std::vector<int>& cdf);
00289 };
00290
00291 #endif

```

4.7 temp/main.cpp File Reference

```

#include <iostream>
#include <string>
#include <sstream>
#include <vector>
#include <limits>
#include "Filter.h"
#include "Slice.h"
#include "Projection.h"
#include "Volume.h"
#include "utils.h"
#include "algorithm"

```

Functions

- int **main** ()

4.7.1 Detailed Description

Group Name: Fibonacci Name: Yibin Gao Github_name: edsml-kg23 Name: Ruihan He Github_name: edsml-rh323 Name: Yu Yin Github_name: acse-yy923 Name: Sara Lakatos Github_name: acse-sl4623 Name: Manawi Kahie Github_name: acse-mk1923 Name: Wenyi Yang Github_name: acse-wy1023

4.8 temp/Projection.cpp File Reference

```

#include "Projection.h"
#include <limits>
#include <iostream>
#include <vector>
#include "utils.h"

```

Functions

- int **adjustIndex** (int userIndex, int sliceCount)

4.8.1 Detailed Description

Group Name: Fibonacci Name: Yibin Gao Github_name: edsml-kg23 Name: Ruihan He Github_name: edsml-rh323 Name: Yu Yin Github_name: acse-yy923 Name: Sara Lakatos Github_name: acse-sl4623 Name: Manawi Kahie Github_name: acse-mk1923 Name: Wenyi Yang Github_name: acse-wy1023

4.9 temp/Projection.h File Reference

```
#include "Image.h"
#include "Volume.h"
```

Classes

- class [Projection](#)

4.9.1 Detailed Description

Group Name: Fibonacci Name: Yibin Gao Github_name: edsml-kg23 Name: Ruihan He Github_name: edsml-rh323 Name: Yu Yin Github_name: acse-yy923 Name: Sara Lakatos Github_name: acse-sl4623 Name: Manawi Kahie Github_name: acse-mk1923 Name: Wenyi Yang Github_name: acse-wy1023

4.10 Projection.h

[Go to the documentation of this file.](#)

```
00001
00013 // Projection.h
00014 #ifndef PROJECTION_H
00015 #define PROJECTION_H
00016 #include "Image.h"
00017 #include "Volume.h"
00018
00019 class Projection {
00020 public:
00035     static Image MIP(const Volume& volume, int minIndex, int maxIndex);
00036
00051     static Image MinIP(const Volume& volume, int minIndex, int maxIndex);
00052
00067     static Image AIP(const Volume& volume, int minIndex, int maxIndex);
00068
00085     static Image AIP_Median(const Volume& volume, int minIndex, int maxIndex);
00086 };
00087
00088 #endif // PROJECTION_H
```

4.11 temp/Slice.cpp File Reference

```
#include "Slice.h"
#include <iostream>
#include "Volume.h"
#include <filesystem>
```

4.11.1 Detailed Description

Group Name: Fibonacci Name: Yibin Gao Github_name: edsml-kg23 Name: Ruihan He Github_name: edsml-rh323 Name: Yu Yin Github_name: acse-yy923 Name: Sara Lakatos Github_name: acse-sl4623 Name: Manawi Kahie Github_name: acse-mk1923 Name: Wenyi Yang Github_name: acse-wy1023

4.12 temp/Slice.h File Reference

```
#include "Volume.h"
#include "Image.h"
#include <string>
```

Classes

- class [Slicer](#)

4.12.1 Detailed Description

Group Name: Fibonacci Name: Yibin Gao Github_name: edsml-kg23 Name: Ruihan He Github_name: edsml-rh323 Name: Yu Yin Github_name: acse-yy923 Name: Sara Lakatos Github_name: acse-sl4623 Name: Manawi Kahie Github_name: acse-mk1923 Name: Wenyi Yang Github_name: acse-wy1023

4.13 Slice.h

[Go to the documentation of this file.](#)

```
00001
00012 //Slice.h
00013 #ifndef SLICE_H
00014 #define SLICE_H
00015
00016 #include "Volume.h"
00017 #include "Image.h"
00018 #include <string>
00019
00020 class Slicer {
00021 public:
00022
00023
00036     static Image sliceXY(const Volume& volume, int z);
00037
00048     static Image sliceYZ(const Volume& volume, int x);
00049
00061     static Image sliceXZ(const Volume& volume, int y);
00062
00063
00078     // Declare the new method for generating slices at specific coordinates
00079
00080     static void generateSlices(const Volume& volume, const std::string& sliceFolder, int xCoordinate,
00081 int yCoordinate);
00082
00083 #endif // SLICE_H
```

4.14 temp/utils.cpp File Reference

```
#include <cmath>
#include <iostream>
#include "Filter.h"
#include "utils.h"
```

Functions

- int [partition](#) (std::vector< unsigned char > &arr, int left, int right)
Partitions the array for the quicksort algorithm.
- unsigned char [quickSelect](#) (std::vector< unsigned char > &arr, int left, int right, int k)
Apply quick select algorithm to find the k-th smallest element.
- std::vector< double > [generate1DGaussianKernel](#) (int kernelSize, double sigma)
Generate 1D Gaussian kernel.
- void [apply1DConvolution2D](#) ([Image](#) &inputImage, [Image](#) &outputImage, const std::vector< double > &kernel, Axis axis)
Applies a 1D convolution to a 2D image along a specified axis.
- void [apply1DConvolution3D](#) ([Volume](#) &inputVolume, [Volume](#) &outputVolume, const std::vector< double > &kernel, Axis axis)
Apply 1D convolution to a volume.
- void [merge](#) (std::vector< std::string > &filePaths, int left, int middle, int right)
Merge two sorted arrays.
- void [mergeSort](#) (std::vector< std::string > &filePaths, int left, int right)
Merge sort the array.
- std::string [extractFilename](#) (const std::string &path)
Extract the filename from the path.
- std::string [extractFilenameWithoutExtension](#) (const std::string &filename)
Extract the filename without extension.
- std::string [convertNumberToString](#) (int number)
Convert number to string.
- void [createOutputDirectory](#) (const std::string &path)
Create output directory.

4.14.1 Detailed Description

Group Name: Fibonacci Name: Yibin Gao Github_name: edsml-kg23 Name: Ruihan He Github_name: edsml-rh323 Name: Yu Yin Github_name: acse-yy923 Name: Sara Lakatos Github_name: acse-sl4623 Name: Manawi Kahie Github_name: acse-mk1923 Name: Wenyi Yang Github_name: acse-wy1023

4.14.2 Function Documentation

4.14.2.1 apply1DConvolution2D()

```
void apply1DConvolution2D (
    Image & inputImage,
    Image & outputImage,
    const std::vector< double > & kernel,
    Axis axis )
```

Applies a 1D convolution to a 2D image along a specified axis.

Parameters

<i>inputImage</i>	The input image to be convolved. This image remains unchanged.
<i>outputImage</i>	The output image where the result of the convolution is stored.
<i>kernel</i>	The convolution kernel as a vector of doubles. The kernel should be normalized.
<i>axis</i>	The axis along which to apply the convolution, specified as an <code>Axis</code> type. This can be <code>Axis::X</code> for horizontal or <code>Axis::Y</code> for vertical convolution.

This function applies a 1D convolution to the input image using the specified kernel. The convolution can be performed either horizontally (along the X axis) or vertically (along the Y axis) as specified by the `axis` parameter. The function handles edge pixels by replicating the nearest edge value, ensuring that the output image has the same dimensions as the input. The output image is then filled with the results of the convolution.

4.14.2.2 apply1DConvolution3D()

```
void apply1DConvolution3D (
    Volume & inputVolume,
    Volume & outputVolume,
    const std::vector< double > & kernel,
    Axis axis )
```

Apply 1D convolution to a volume.

Parameters

<i>inputVolume</i>	Input volume for convolution.
<i>outputVolume</i>	<code>Volume</code> to store the convolved data.
<i>kernel</i>	Convolution kernel, normalized to ensure intensity preservation.
<i>axis</i>	Axis along which the convolution is applied (X, Y, or Z).

Performs 1D convolution across one axis of a 3D volume with a specified kernel. It's used to apply filters like Gaussian blur in a separable manner. The function ensures edges are handled by replicating edge values, maintaining the volume's dimensions.

4.14.2.3 convertNumberToString()

```
std::string convertNumberToString (
    int number )
```

Convert number to string.

Parameters

<i>number</i>	input number
---------------	--------------

Returns

`std::string` resultant string

4.14.2.4 createOutputDirectory()

```
void createOutputDirectory (
    const std::string & path )
```

Create output directory.

Parameters

<i>path</i>	input path
-------------	------------

4.14.2.5 extractFilename()

```
std::string extractFilename (
    const std::string & path )
```

Extract the filename from the path.

Parameters

<i>path</i>	input path
-------------	------------

Returns

std::string resultant filename

4.14.2.6 extractFilenameWithoutExtension()

```
std::string extractFilenameWithoutExtension (
    const std::string & filename )
```

Extract the filename without extension.

Parameters

<i>filename</i>	input filename
-----------------	----------------

Returns

std::string resultant filename

4.14.2.7 generate1DGaussianKernel()

```
std::vector< double > generate1DGaussianKernel (
    int kernelSize,
    double sigma )
```

Generate 1D Gaussian kernel.

Parameters

<i>kernelSize</i>	The size of the kernel. It determines the length of the generated Gaussian kernel vector. The kernel size should be a positive odd integer to ensure symmetry around the center.
<i>sigma</i>	The standard deviation of the Gaussian distribution.

Returns

`std::vector<double>` A vector of doubles representing the normalized 1D Gaussian kernel.

This function generates a 1D Gaussian kernel with a specified size and standard deviation (*sigma*). The generated kernel is normalized so that the sum of its elements equals 1, ensuring that the application of the kernel does not change the overall brightness of the image.

4.14.2.8 merge()

```
void merge (
    std::vector< std::string > & filePaths,
    int left,
    int middle,
    int right )
```

Merge two sorted arrays.

Parameters

<i>arr</i>	input array
<i>left</i>	input left index
<i>middle</i>	input middle index
<i>right</i>	input right index

4.14.2.9 mergeSort()

```
void mergeSort (
    std::vector< std::string > & filePaths,
    int left,
    int right )
```

Merge sort the array.

Parameters

<i>arr</i>	input array
<i>left</i>	input left index
<i>right</i>	input right index

4.14.2.10 partition()

```
int partition (
    std::vector< unsigned char > & arr,
    int left,
    int right )
```

Partitions the array for the quicksort algorithm.

Parameters

<i>arr</i>	Reference to the vector of unsigned char elements to be partitioned.
<i>left</i>	The starting index of the segment of the array to be partitioned.
<i>right</i>	The ending index of the segment of the array to be partitioned.

Returns

int The index position of the pivot element after partitioning.

This function takes the last element as pivot, places the pivot element at its correct position in the sorted array, and places all smaller (smaller than pivot) to the left of the pivot and all greater elements to the right of the pivot. This is a helper function used by the quicksort algorithm for sorting elements within an array. Reference: Quickselect algorithm: <https://chat.openai.com/share/4db4258a-0cc1-4d63-b2fd-8191371133e1>

4.14.2.11 quickSelect()

```
unsigned char quickSelect (
    std::vector< unsigned char > & arr,
    int left,
    int right,
    int k )
```

Apply quick select algorithm to find the k-th smallest element.

Parameters

<i>arr</i>	Reference to the vector of unsigned char elements from which to select.
<i>left</i>	The starting index of the segment within the array to perform the selection.
<i>right</i>	The ending index of the segment within the array to perform the selection.
<i>k</i>	The index (0-based) of the smallest element to find. For example, k = 0 will find the smallest element, and k = length of array - 1 will find the largest element.

Returns

unsigned char The k-th smallest element in the specified segment of the array.

Quickselect is a selection algorithm to find the k-th smallest element in an unordered list. Quickselect uses the same overall approach as quicksort, choosing one element as a pivot and partitioning the data in two based on the pivot, accordingly as less than or greater than the pivot.

4.15 temp/utils.h File Reference

```
#include <iostream>
#include <vector>
#include <cmath>
#include "Image.h"
#include "Volume.h"
#include "Filter.h"
```

Functions

- int [partition](#) (std::vector< unsigned char > &arr, int left, int right)
Partitions the array for the quicksort algorithm.
- unsigned char [quickSelect](#) (std::vector< unsigned char > &arr, int left, int right, int k)
Apply quick select algorithm to find the k-th smallest element.
- std::vector< double > [generate1DGaussianKernel](#) (int kernelSize, double sigma)
Generate 1D Gaussian kernel.
- void [apply1DConvolution2D](#) ([Image](#) &inputImage, [Image](#) &outputImage, const std::vector< double > &kernel, Axis axis)
Applies a 1D convolution to a 2D image along a specified axis.
- void [apply1DConvolution3D](#) ([Volume](#) &inputVolume, [Volume](#) &outputVolume, const std::vector< double > &kernel, Axis axis)
Apply 1D convolution to a volume.
- void [merge](#) (std::vector< std::string > &filePaths, int left, int middle, int right)
Merge two sorted arrays.
- void [mergeSort](#) (std::vector< std::string > &filePaths, int left, int right)
Merge sort the array.
- std::string [extractFilenameWithoutExtension](#) (const std::string &filename)
Extract the filename without extension.
- std::string [convertNumberToString](#) (int number)
Convert number to string.
- void [createOutputDirectory](#) (const std::string &path)
Create output directory.
- std::string [extractFilename](#) (const std::string &path)
Extract the filename from the path.

4.15.1 Detailed Description

Group Name: Fibonacci Name: Yibin Gao Github_name: edsml-kg23 Name: Ruihan He Github_name: edsml-rh323 Name: Yu Yin Github_name: acse-yy923 Name: Sara Lakatos Github_name: acse-sl4623 Name: Manawi Kahie Github_name: acse-mk1923 Name: Wenyi Yang Github_name: acse-wy1023

4.15.2 Function Documentation

4.15.2.1 apply1DConvolution2D()

```
void apply1DConvolution2D (
    Image & inputImage,
    Image & outputImage,
    const std::vector< double > & kernel,
    Axis axis )
```

Applies a 1D convolution to a 2D image along a specified axis.

Parameters

<i>inputImage</i>	The input image to be convolved. This image remains unchanged.
<i>outputImage</i>	The output image where the result of the convolution is stored.
<i>kernel</i>	The convolution kernel as a vector of doubles. The kernel should be normalized.
<i>axis</i>	The axis along which to apply the convolution, specified as an <code>Axis</code> type. This can be <code>Axis::X</code> for horizontal or <code>Axis::Y</code> for vertical convolution.

This function applies a 1D convolution to the input image using the specified kernel. The convolution can be performed either horizontally (along the X axis) or vertically (along the Y axis) as specified by the `axis` parameter. The function handles edge pixels by replicating the nearest edge value, ensuring that the output image has the same dimensions as the input. The output image is then filled with the results of the convolution.

4.15.2.2 `apply1DConvolution3D()`

```
void apply1DConvolution3D (
    Volume & inputVolume,
    Volume & outputVolume,
    const std::vector< double > & kernel,
    Axis axis )
```

Apply 1D convolution to a volume.

Parameters

<i>inputVolume</i>	Input volume for convolution.
<i>outputVolume</i>	<code>Volume</code> to store the convolved data.
<i>kernel</i>	Convolution kernel, normalized to ensure intensity preservation.
<i>axis</i>	Axis along which the convolution is applied (X, Y, or Z).

Performs 1D convolution across one axis of a 3D volume with a specified kernel. It's used to apply filters like Gaussian blur in a separable manner. The function ensures edges are handled by replicating edge values, maintaining the volume's dimensions.

4.15.2.3 `convertNumberToString()`

```
std::string convertNumberToString (
    int number )
```

Convert number to string.

Parameters

<i>number</i>	input number
---------------	--------------

Returns

`std::string` resultant string

4.15.2.4 createOutputDirectory()

```
void createOutputDirectory (
    const std::string & path )
```

Create output directory.

Parameters

<i>path</i>	input path
-------------	------------

4.15.2.5 extractFilename()

```
std::string extractFilename (
    const std::string & path )
```

Extract the filename from the path.

Parameters

<i>path</i>	input path
-------------	------------

Returns

std::string resultant filename

4.15.2.6 extractFilenameWithoutExtension()

```
std::string extractFilenameWithoutExtension (
    const std::string & filename )
```

Extract the filename without extension.

Parameters

<i>filename</i>	input filename
-----------------	----------------

Returns

std::string resultant filename

4.15.2.7 generate1DGaussianKernel()

```
std::vector< double > generate1DGaussianKernel (
    int kernelSize,
    double sigma )
```

Generate 1D Gaussian kernel.

Parameters

<i>kernelSize</i>	The size of the kernel. It determines the length of the generated Gaussian kernel vector. The kernel size should be a positive odd integer to ensure symmetry around the center.
<i>sigma</i>	The standard deviation of the Gaussian distribution.

Returns

`std::vector<double>` A vector of doubles representing the normalized 1D Gaussian kernel.

This function generates a 1D Gaussian kernel with a specified size and standard deviation (sigma). The generated kernel is normalized so that the sum of its elements equals 1, ensuring that the application of the kernel does not change the overall brightness of the image.

4.15.2.8 merge()

```
void merge (
    std::vector< std::string > & filePaths,
    int left,
    int middle,
    int right )
```

Merge two sorted arrays.

Parameters

<i>arr</i>	input array
<i>left</i>	input left index
<i>middle</i>	input middle index
<i>right</i>	input right index

4.15.2.9 mergeSort()

```
void mergeSort (
    std::vector< std::string > & filePaths,
    int left,
    int right )
```

Merge sort the array.

Parameters

<i>arr</i>	input array
<i>left</i>	input left index
<i>right</i>	input right index

4.15.2.10 partition()

```
int partition (
    std::vector< unsigned char > & arr,
    int left,
    int right )
```

Partitions the array for the quicksort algorithm.

Parameters

<i>arr</i>	Reference to the vector of unsigned char elements to be partitioned.
<i>left</i>	The starting index of the segment of the array to be partitioned.
<i>right</i>	The ending index of the segment of the array to be partitioned.

Returns

int The index position of the pivot element after partitioning.

This function takes the last element as pivot, places the pivot element at its correct position in the sorted array, and places all smaller (smaller than pivot) to the left of the pivot and all greater elements to the right of the pivot. This is a helper function used by the quicksort algorithm for sorting elements within an array. Reference: Quickselect algorithm: <https://chat.openai.com/share/4db4258a-0cc1-4d63-b2fd-8191371133e1>

4.15.2.11 quickSelect()

```
unsigned char quickSelect (
    std::vector< unsigned char > & arr,
    int left,
    int right,
    int k )
```

Apply quick select algorithm to find the k-th smallest element.

Parameters

<i>arr</i>	Reference to the vector of unsigned char elements from which to select.
<i>left</i>	The starting index of the segment within the array to perform the selection.
<i>right</i>	The ending index of the segment within the array to perform the selection.
<i>k</i>	The index (0-based) of the smallest element to find. For example, k = 0 will find the smallest element, and k = length of array - 1 will find the largest element.

Returns

unsigned char The k-th smallest element in the specified segment of the array.

Quickselect is a selection algorithm to find the k-th smallest element in an unordered list. Quickselect uses the same overall approach as quicksort, choosing one element as a pivot and partitioning the data in two based on the pivot, accordingly as less than or greater than the pivot.

4.16 utils.h

[Go to the documentation of this file.](#)

```

00001
00013 #ifndef UTILS_H
00014 #define UTILS_H
00015
00016 #include <iostream>
00017 #include <vector>
00018 #include <cmath>
00019 #include <vector>
00020 #include "Image.h" // Make sure this is correctly included for Image class definition
00021 #include "Volume.h" // Include the header file that defines the "Volume" class
00022 #include "Filter.h" // Include if necessary for Axis enum
00023
00039 int partition(std::vector<unsigned char>& arr, int left, int right);
00040
00055 unsigned char quickSelect(std::vector<unsigned char>& arr, int left, int right, int k);
00056
00069 std::vector<double> generate1DGaussianKernel(int kernelSize, double sigma);
00070
00086 void apply1DConvolution2D(Image& inputImage, Image& outputImage, const std::vector<double>& kernel,
    Axis axis);
00087
00100 void apply1DConvolution3D(Volume& inputVolume, Volume& outputVolume, const std::vector<double>&
    kernel, Axis axis);
00101
00110 void merge(std::vector<std::string>& filePaths, int left, int middle, int right);
00111
00119 void mergeSort(std::vector<std::string>& filePaths, int left, int right);
00120
00127 std::string extractFilenameWithoutExtension(const std::string &filename);
00128
00135 std::string convertNumberToString(int number);
00136
00142 void createOutputDirectory(const std::string &path);
00143
00150 std::string extractFilename(const std::string &path);
00151
00152 #endif // UTILS_H
00153

```

4.17 temp/Volume.cpp File Reference

```

#include "Volume.h"
#include <iostream>
#include <filesystem>
#include <string>
#include <sstream>
#include <algorithm>
#include "utils.h"

```

4.17.1 Detailed Description

Group Name: Fibonacci Name: Yibin Gao Github_name: edsml-kg23 Name: Ruihan He Github_name: edsml-rh323 Name: Yu Yin Github_name: acse-yy923 Name: Sara Lakatos Github_name: acse-sl4623 Name: Manawi Kahie Github_name: acse-mk1923 Name: Wenyi Yang Github_name: acse-wy1023

4.18 temp/Volume.h File Reference

```

#include "Image.h"
#include <vector>
#include <string>

```

Classes

- class [Volume](#)

4.18.1 Detailed Description

Group Name: Fibonacci Name: Yibin Gao Github_name: edsml-kg23 Name: Ruihan He Github_name: edsml-rh323 Name: Yu Yin Github_name: acse-yy923 Name: Sara Lakatos Github_name: acse-sl4623 Name: Manawi Kahie Github_name: acse-mk1923 Name: Wenyi Yang Github_name: acse-wy1023

4.19 Volume.h

[Go to the documentation of this file.](#)

```

00001
00013 // Volume.h
00014 #ifndef VOLUME_H
00015 #define VOLUME_H
00016
00017 #include "Image.h"
00018 #include <vector>
00019 #include <string>
00020
00021 class Volume {
00022 public:
00027     Volume(); // use the directory path to load the volume
00028
00034     Volume(const std::string& directoryPath);
00035
00042     Image getSlice(int index) const; // get the image slice from the volume index
00043
00049     void loadVolume(const std::string& directoryPath); // load the volume from the directory path
00050
00058     void createEmpty(int width, int height, int depth);
00059
00065     int getWidth() const;
00066
00072     int getHeight() const;
00073
00079     int getDepth() const;
00080
00089     unsigned char getVoxel(int x, int y, int z) const;
00090
00099     void setVoxel(unsigned char v, int x, int y, int z);
00100
00107     void updateSlice(int z, Image newData);
00108
00114     void saveVolume(const std::string& outputDirectory) const;
00115
00116     // Added methods
00123     Image sliceXY(int zCoordinate) const;
00124
00131     Image sliceXZ(int yCoordinate) const;
00132
00139     Image sliceYZ(int xCoordinate) const;
00140
00146     // Inline definition of getSlices
00147     const std::vector<Image>& getSlices() const { return slices; }
00148
00149 private:
00150     int width, height, depth;
00151     std::vector<Image> slices; // store the slices of the volume
00152 };
00153
00154 #endif

```


Index

- addSaltPepperNoise
 - Filter, [5](#)
- AIP
 - Projection, [19](#)
- AIP_Median
 - Projection, [19](#)
- apply1DConvolution2D
 - utils.cpp, [38](#)
 - utils.h, [43](#)
- apply1DConvolution3D
 - utils.cpp, [39](#)
 - utils.h, [44](#)
- applyBoxFilter2D
 - Filter, [6](#)
- applyGaussianFilter2D
 - Filter, [6](#)
- applyGaussianFilter3D
 - Filter, [6](#)
- applyMedianFilter2D
 - Filter, [7](#)
- applyMedianFilter3D
 - Filter, [7](#)
- brightness
 - Filter, [7](#)
- calculateCdf
 - Image, [10](#)
- clone
 - Image, [10](#)
- convertNumberToString
 - utils.cpp, [39](#)
 - utils.h, [44](#)
- createEmpty
 - Image, [11](#)
 - Volume, [25](#)
- createOutputDirectory
 - utils.cpp, [39](#)
 - utils.h, [44](#)
- extractFilename
 - utils.cpp, [40](#)
 - utils.h, [45](#)
- extractFilenameWithoutExtension
 - utils.cpp, [40](#)
 - utils.h, [45](#)
- Filter, [5](#)
 - addSaltPepperNoise, [5](#)
 - applyBoxFilter2D, [6](#)
 - applyGaussianFilter2D, [6](#)
 - applyGaussianFilter3D, [6](#)
 - applyMedianFilter2D, [7](#)
 - applyMedianFilter3D, [7](#)
 - brightness, [7](#)
 - grayscale, [8](#)
 - histogramEqualization, [8](#)
 - threshold, [8](#)
- generate1DGaussianKernel
 - utils.cpp, [40](#)
 - utils.h, [45](#)
- generateSlices
 - Slicer, [22](#)
- getDepth
 - Volume, [26](#)
- getFileName
 - Image, [11](#)
- getHeight
 - Image, [11](#)
 - Volume, [26](#)
- getPixel
 - Image, [12](#)
- getSlice
 - Volume, [26](#)
- getSlices
 - Volume, [26](#)
- getVoxel
 - Volume, [27](#)
- getWidth
 - Image, [13](#)
 - Volume, [27](#)
- grayscale
 - Filter, [8](#)
- histogramEqualization
 - Filter, [8](#)
- hslToRgb
 - Image, [13](#)
- hsvToRgb
 - Image, [14](#)
- Image, [9](#)
 - calculateCdf, [10](#)
 - clone, [10](#)
 - createEmpty, [11](#)
 - getFileName, [11](#)
 - getHeight, [11](#)
 - getPixel, [12](#)
 - getWidth, [13](#)

- hslToRgb, 13
- hsvToRgb, 14
- isSameAs, 14
- load, 15
- operator-, 15
- rgbToHsl, 16
- rgbToHsv, 16
- save, 17
- setPixel, 17, 18
- isSameAs
 - Image, 14
- load
 - Image, 15
- loadVolume
 - Volume, 27
- merge
 - utils.cpp, 41
 - utils.h, 46
- mergeSort
 - utils.cpp, 41
 - utils.h, 46
- MinIP
 - Projection, 20
- MIP
 - Projection, 20
- operator-
 - Image, 15
- partition
 - utils.cpp, 41
 - utils.h, 46
- Projection, 19
 - AIP, 19
 - AIP_Median, 19
 - MinIP, 20
 - MIP, 20
- quickSelect
 - utils.cpp, 42
 - utils.h, 47
- rgbToHsl
 - Image, 16
- rgbToHsv
 - Image, 16
- save
 - Image, 17
- saveVolume
 - Volume, 27
- setPixel
 - Image, 17, 18
- setVoxel
 - Volume, 28
- Slicer, 21
 - generateSlices, 22
 - sliceXY, 23
 - sliceXZ, 23
 - sliceYZ, 24
- sliceXY
 - Slicer, 23
 - Volume, 28
- sliceXZ
 - Slicer, 23
 - Volume, 28
- sliceYZ
 - Slicer, 24
 - Volume, 29
- temp/Filter.cpp, 31
- temp/Filter.h, 31, 32
- temp/Image.cpp, 33
- temp/Image.h, 34
- temp/main.cpp, 35
- temp/Projection.cpp, 35
- temp/Projection.h, 36
- temp/Slice.cpp, 36
- temp/Slice.h, 37
- temp/utils.cpp, 38
- temp/utils.h, 43, 48
- temp/Volume.cpp, 48
- temp/Volume.h, 48, 49
- threshold
 - Filter, 8
- updateSlice
 - Volume, 29
- utils.cpp
 - apply1DConvolution2D, 38
 - apply1DConvolution3D, 39
 - convertNumberToString, 39
 - createOutputDirectory, 39
 - extractFilename, 40
 - extractFilenameWithoutExtension, 40
 - generate1DGaussianKernel, 40
 - merge, 41
 - mergeSort, 41
 - partition, 41
 - quickSelect, 42
- utils.h
 - apply1DConvolution2D, 43
 - apply1DConvolution3D, 44
 - convertNumberToString, 44
 - createOutputDirectory, 44
 - extractFilename, 45
 - extractFilenameWithoutExtension, 45
 - generate1DGaussianKernel, 45
 - merge, 46
 - mergeSort, 46
 - partition, 46
 - quickSelect, 47
- Volume, 24
 - createEmpty, 25
 - getDepth, 26
 - getHeight, 26

[getSlice](#), [26](#)
[getSlices](#), [26](#)
[getVoxel](#), [27](#)
[getWidth](#), [27](#)
[loadVolume](#), [27](#)
[saveVolume](#), [27](#)
[setVoxel](#), [28](#)
[sliceXY](#), [28](#)
[sliceXZ](#), [28](#)
[sliceYZ](#), [29](#)
[updateSlice](#), [29](#)
[Volume](#), [25](#)