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	
4.1.1 Detailed Description	
4.2 temp/Filter.h File Reference	
4.2.1 Detailed Description	
4.3 Filter.h	
4.4 temp/Image.cpp File Reference	
4.4.1 Detailed Description	
4.5 temp/Image.h File Reference	
4.5.1 Detailed Description	
4.6 Image.h	
4.7 temp/main.cpp File Reference	
4.7.1 Detailed Description	
4.8 temp/Projection.cpp File Reference	
4.8.1 Detailed Description	

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

Index 51

Chapter 1

Class Index

1.1 Class List

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

ter	Ę
nage	Ş
ojection	19
icer	21
Numa	2/

2 Class Index

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/lmage.cpp	
temp/lmage.h	34
temp/main.cpp	35
temp/Projection.cpp	35
temp/Projection.h	
temp/Slice.cpp	
temp/Slice.h	
temp/utils.cpp	
temp/utils.h	
temp/Volume.cpp	
temp/Volume.h	48

File Index

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()

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

Parameters

image	The image to add noise to.
noisePercentage	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()

Applies a box blur filter to the given image.

Parameters

image	The image to apply the filter to.
kernel_size	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()

Applies a Gaussian blur filter to the given image.

Parameters

image	The image to apply the filter to.
kernel_size	The size of the kernel used for blurring.
sigma	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()

Apply a Guassian filter to a 3d volume.

3.1 Filter Class Reference 7

Parameters

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

Applies the image median blur filter to the given image.

Parameters

image	The image to apply the filter to.
kernel_size	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: $https://chat.openai. \leftarrow com/share/bfc5caa6-324f-4f8a-9967-d4a960433151$

3.1.1.6 applyMedianFilter3D()

Apply a median filter to a 3d volume.

Parameters

volume	Reference to the Volume object to be filtered. The volume is updated in-place with the filtered results.
kernelSize	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()

Applies a brightness filter to the given image.

Parameters

image The image to apply the filter to.	
brightness	The brightness value to apply to the image.
autoBrightness	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()

Applies a grayscale filter to the given image.

Parameters

image	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: gray = 0.2126 * r + 0.7152 * g + 0.0722 * b

3.1.1.9 histogramEqualization()

Applies a histogram equalization filter to the given image.

Parameters

image	The image to apply the filter to.
hsv	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()

Applies a threshold filter to the given image.

Parameters

image	The image to apply the filter to.
threshold	The threshold value to use for binarization.
useHsv	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 rgbToHsI (float r, float g, float b, float &h, float &s, float &l)
 rgb to hsl conversion
- static void hslToRgb (float h, float s, float I, 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()

calculate cdf

Parameters

histogram	input histogram
cdf	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()

Create a Empty object.

Parameters

width	input width
height	input height
channels	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]

Get the Pixel object.

Parameters

X	input x coordinate
У	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]

Get the Pixel object.

Parameters

X	input x coordinate
У	input y coordinate
ch	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

h	input hue value
s	input saturation value
1	input lightness value
r	output red value
Gener	ated by Doxygen
g	ated by Doxygen output green value
b	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

h	input hue value
s	input saturation value
V	input value
r	output red value
g	output green value
b	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()

check if the image is same as the other image

Parameters

```
other 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()

load the image

Parameters

```
path 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-()

· operator overloading

Parameters

other	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 rgbToHsI()

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

rgb to hsl conversion

Parameters

r	input red value
g	input green value
b	input blue value
h	output hue value
s	output saturation value
1	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()

rgb to hsv conversion

Parameters

r	input red value
g	input green value
b	input blue value
h	output hue value
s	output saturation value
V	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()

save the image

Parameters

```
path 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

х	input x coordinate
У	input y coordinate
ch	input channel
value	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

X	input x coordinate
У	input y coordinate
value	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/lmage.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()

Apply average intensity projection to 3d volume.

Parameters

volume	input volume
minIndex	input min index
maxIndex	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()

Apply median intensity projection to 3d volume.

Parameters

volume	input volume
minIndex	input min index
maxIndex	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()

Apply min intensity projection to 3d volume.

Parameters

volume	input volume
minIndex	input min index
maxIndex	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

3.4 Slicer Class Reference 21

3.3.1.4 MIP()

Apply max intensity projection to 3d volume.

Parameters

volume	input volume
minIndex	input min index
maxIndex	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 y
 —
 Coordinate)

Generate slices at specific coordinates.

3.4.1 Member Function Documentation

3.4.1.1 generateSlices()

Generate slices at specific coordinates.

3.4 Slicer Class Reference 23

Parameters

volume	input volume
sliceFolder	input slice folder
xCoordinate	input x coordinate
yCoordinate	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

volume	input volume
Z	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

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

Generated by Doxygen

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()

Slice the volume in YZ plane at x.

Parameters

volume	input volume
X	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()

Construct a new Volume object from the directory path.

Parameters

```
directoryPath input directory path
```

3.5.2 Member Function Documentation

3.5.2.1 createEmpty()

Create a Empty object.

Parameters

width	input width
height	input height
depth	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()

Get the Slice object.

Parameters

|--|

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()

Get the Voxel object.

Parameters

X	input x
У	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()

load the volume from the directory path

Parameters

directoryPath	input directory path

3.5.2.9 saveVolume()

save the volume to the output directory

Parameters

```
outputDirectory 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

V	input v
Х	input x
У	input y
Z	input z

3.5.2.11 sliceXY()

slice the volume in the XY plane

Parameters

zCoordinate	input z coordinate
-------------	--------------------

Returns

Image resultant image

3.5.2.12 sliceXZ()

slice the volume in the XZ plane

Parameters

yCoordinate	input y coordinate
-------------	--------------------

Returns

Image resultant image

3.5.2.13 sliceYZ()

slice the volume in the YZ plane

Parameters

<i>xCoordinate</i> ir	nput x coordinate
-----------------------	-------------------

Returns

Image resultant image

3.5.2.14 updateSlice()

```
void Volume::updateSlice (  \mbox{int $z$,} \\ \mbox{Image $newData$} \mbox{)}
```

Update the Slice object.

Parameters

Z	input z
newData	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
               Reflect, // Reflect padding
Copy // Copy edge padding
00028
00029
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 * 00037 * @param gx
                       The function directly modifies the image object to store the result.
                        The horizontal gradient operator (kernel) for edge detection.
00038 * @param gy
                       The vertical gradient operator (kernel) for edge detection.
00039
      \star @param padding Enum value specifying the padding method to use (Zero, Reflect, Copy).
00040
00041
00042 \, \star - Computes horizontal and vertical gradients at each pixel using gx and gy kernels.
00043 \star - Calculates the gradient magnitude as the Euclidean norm of the horizontal and vertical gradients. 00044 \star - Normalizes the gradient magnitude to fit within the range of 0 to 255.
        - Supports different padding methods to handle edge pixels.
       * - reference: https://chat.openai.com/share/6133745c-f97e-465e-a3a8-a39cfe72eeaa
00046
00047
00048
          static void applyEdgeDetection(Image &image, const int gx[3][3], const int gy[3][3],
                                            PaddingMethod padding = PaddingMethod::Reflect);
00049
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
          static void addSaltPepperNoise(Image &image, float noisePercentage);
00111
00112
00126
          static void applyMedianFilter2D(Image& image, int kernel_size);
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 \star @param image Reference to an Image object that will be processed.
                      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
00164
          static void sobel (Image &image);
00165
00166 /*
00167 * Applies the Robert Cross edge detection filter to an image.
00168 *
00169 \star @param image Reference to an Image object that will be processed.
00170 *
                      The result is stored directly in this object.
00171 * Process:
00173 \star - 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.
00182 \,\,\star\, @param image Reference to an Image object that will be processed.
00183 *
                      The result is stored directly in this object.
00184 *
00185 * Process:
00186 \,\star\, - Uses predefined Scharr operators for more accurate horizontal and vertical gradient calculation.
00187 \,\star\, - 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 \,\star\, - Uses predefined Prewitt operators for horizontal and vertical gradient calculation.
00200 \,\star\, - 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);
00234
          static void applyMedianFilter3D(Volume& volume, int kernelSize);
00235
00236 };
00237 #endif
00238
```

4.4 temp/lmage.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-kg23 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/lmage.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 TMAGE H
00015 #define IMAGE_H
00016
00017 #include <string>
00018 #include <vector>
00019
00020 class Image {
00021 public:
          //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
          Image(int width = 0, int height = 0, int channels = 3);
00163
00164
          // getter functions
00165
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
          // Utility functions declarations
00196
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);
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
08000
          static void generateSlices(const Volume& volume, const std::string& sliceFolder, int xCoordinate,
      int yCoordinate);
00081 };
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()

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

Parameters

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

Apply 1D convolution to a volume.

Parameters

inputVolume Input volume for convolution.		
outputVolume	Volume to store the convolved data.	
kernel Convolution kernel, normalized to ensure intensity preservation.		
axis 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()

Convert number to string.

Parameters

number	input number

Returns

std::string resultant string

4.14.2.4 createOutputDirectory()

```
void createOutputDirectory ( {\tt const\ std::string\ \&\ path\ )}
```

Create output directory.

Parameters

```
path input path
```

4.14.2.5 extractFilename()

Extract the filename from the path.

Parameters

```
path input path
```

Returns

std::string resultant filename

4.14.2.6 extractFilenameWithoutExtension()

Extract the filename without extension.

Parameters

```
filename input filename
```

Returns

std::string resultant filename

4.14.2.7 generate1DGaussianKernel()

Generate 1D Gaussian kernel.

Parameters

kernelSize	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.	
sigma	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

arr	input array
left	input left index
middle	input middle index
right	input right index

4.14.2.9 mergeSort()

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

Merge sort the array.

Parameters

arr	input array
left	input left index
right	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

arr	Reference to the vector of unsigned char elements to be partitioned.
left	The starting index of the segment of the array to be partitioned.
right	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

arr	Reference to the vector of unsigned char elements from which to select.	
left	The starting index of the segment within the array to perform the selection.	
right	The ending index of the segment within the array to perform the selection.	
k	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()

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

Parameters

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

Apply 1D convolution to a volume.

Parameters

inputVolume	Input volume for convolution.	
outputVolume	Volume to store the convolved data.	
kernel Convolution kernel, normalized to ensure intensity preservation.		
axis 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()

Convert number to string.

Parameters

number	input number

Returns

std::string resultant string

4.15.2.4 createOutputDirectory()

```
void createOutputDirectory ( {\tt const\ std::string\ \&\ path\ )}
```

Create output directory.

Parameters

```
path input path
```

4.15.2.5 extractFilename()

Extract the filename from the path.

Parameters

```
path input path
```

Returns

std::string resultant filename

4.15.2.6 extractFilenameWithoutExtension()

Extract the filename without extension.

Parameters

```
filename 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

kernelSize	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.	
sigma	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

arr	input array
left	input left index
middle	input middle index
right	input right index

4.15.2.9 mergeSort()

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

Merge sort the array.

Parameters

arr	input array
left	input left index
right	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

arr	Reference to the vector of unsigned char elements to be partitioned.
left	The starting index of the segment of the array to be partitioned.
right	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

arr	Reference to the vector of unsigned char elements from which to select.
left	The starting index of the segment within the array to perform the selection.
right	The ending index of the segment within the array to perform the selection.
k	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.

```
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);
00055 unsigned char quickSelect(std::vector<unsigned char>& arr, int left, int right, int k);
00056
00069 std::vector<double> generatelDGaussianKernel(int kernelSize, double sigma);
00070
00086 void apply1DConvolution2D(Image& inputImage, Image& outputImage, const std::vector<double>& kernel,
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);
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>
```

4.19 Volume.h 49

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.

```
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:
          Volume(); // use the directory path to load the volume
00027
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
          // Added methods
00116
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;
          std::vector<Image> slices; // store the slices of the volume
00151
00152 };
00153
00154 #endif
```

Index

addSaltPepperNoise	applyGaussianFilter2D,
Filter, 5	applyGaussianFilter3D,
AIP	applyMedianFilter2D, 7
Projection, 19	applyMedianFilter3D, 7
AIP_Median	brightness, 7
Projection, 19	grayscale, 8
apply1DConvolution2D	histogramEqualization, 8
utils.cpp, 38	threshold, 8
utils.h, 43	
apply1DConvolution3D	generate1DGaussianKernel
utils.cpp, 39	utils.cpp, 40
utils.h, 44	utils.h, 45
applyBoxFilter2D	generateSlices
Filter, 6	Slicer, 22
applyGaussianFilter2D	getDepth
Filter, 6	Volume, 26
applyGaussianFilter3D	getFileName
Filter, 6	Image, 11
applyMedianFilter2D	getHeight
Filter, 7	Image, 11
applyMedianFilter3D	Volume, 26
Filter, 7	getPixel
	Image, 12
brightness	getSlice
Filter, 7	Volume, 26
	getSlices
calculateCdf	Volume, 26
Image, 10	getVoxel
clone	Volume, 27
Image, 10	getWidth
convertNumberToString	Image, 13
utils.cpp, 39	Volume, 27
utils.h, 44	grayscale
createEmpty	Filter, 8
Image, 11	1 1101, 0
Volume, 25	histogramEqualization
createOutputDirectory	Filter, 8
utils.cpp, 39	hslToRgb
utils.h, 44	Image, 13
,	hsvToRgb
extractFilename	Image, 14
utils.cpp, 40	290,
utils.h, 45	Image, 9
extractFilenameWithoutExtension	calculateCdf, 10
utils.cpp, 40	clone, 10
utils.h, 45	createEmpty, 11
, -	getFileName, 11
Filter, 5	getHeight, 11
addSaltPepperNoise, 5	getPixel, 12
applyBoxFilter2D, 6	getWidth, 13
	gottidii, io

52 INDEX

hslToRgb, 13	sliceXZ, 23
hsvToRgb, 14	sliceYZ, 24
isSameAs, 14	sliceXY
load, 15	Slicer, 23
operator-, 15	Volume, 28
rgbToHsl, 16	sliceXZ
rgbToHsv, 16	Slicer, 23
save, 17	Volume, 28
setPixel, 17, 18	sliceYZ
isSameAs	Slicer, 24
Image, 14	Volume, 29
image, 14	volume, 29
load	temp/Filter.cpp, 31
Image, 15	temp/Filter.h, 31, 32
loadVolume	temp/Image.cpp, 33
Volume, 27	temp/lmage.h, 34
merge	temp/main.cpp, 35
utils.cpp, 41	temp/Projection.cpp, 35
• •	temp/Projection.h, 36
utils.h, 46	temp/Slice.cpp, 36
mergeSort	temp/Slice.h, 37
utils.cpp, 41	temp/utils.cpp, 38
utils.h, 46	temp/utils.h, 43, 48
MinIP	temp/Volume.cpp, 48
Projection, 20	temp/Volume.h, 48, 49
MIP	threshold
Projection, 20	Filter, 8
operator-	updateSlice
Image, 15	Volume, 29
partition	utils.cpp
utils.cpp, 41	apply1DConvolution2D, 38
utils.h, 46	apply1DConvolution3D, 39
Projection, 19	convertNumberToString, 39
AIP, 19	createOutputDirectory, 39
AIP_Median, 19	extractFilename, 40
MinIP, 20	extractFilenameWithoutExtension, 40
MIP, 20	generate1DGaussianKernel, 40
IVIIF, 20	merge, 41
quickSelect	mergeSort, 41
utils.cpp, 42	partition, 41
• •	quickSelect, 42
utils.h, 47	utils.h
rgbToHsl	apply1DConvolution2D, 43
-	apply1DConvolution3D, 44
Image, 16	convertNumberToString, 44
rgbToHsv	createOutputDirectory, 44
Image, 16	extractFilename, 45
2012	extractFilenameWithoutExtension, 45
save	
Image, 17	generate1DGaussianKernel, 45
saveVolume	merge, 46
Volume, 27	mergeSort, 46
setPixel	partition, 46
Image, 17, 18	quickSelect, 47
setVoxel	
Volume, 28	Volume, 24
Slicer, 21	createEmpty, 25
generateSlices, 22	getDepth, 26
sliceXY, 23	getHeight, 26
- , -	

INDEX 53

```
getSlice, 26
getSlices, 26
getVoxel, 27
getWidth, 27
loadVolume, 27
saveVolume, 27
setVoxel, 28
sliceXY, 28
sliceXZ, 29
updateSlice, 29
Volume, 25
```