ImagePro

Generated by Doxygen 1.9.4

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 Database Class Reference

Public Member Functions

- Database ()

  Create a database storing all the images.
- void load (const std::string &)

  Load database using files sotred in a specified directory.
- std::vector< Image > getImages () const

  Get all the images stored in the database.
- Image getImageById (const int)

  Get an image with a specific id.
- void setImages (const std::vector< Image > &)

  Define the images in database.
- void addImage (const Image &)

  Add new images in the database.

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Database()

```
Database::Database ( )
```

Create a database storing all the images.

A vector storing all images

3.1.2 Member Function Documentation

3.1.2.1 addImage()

```
void Database::addImage (
            const Image & image )
```

Add new images in the database.

Parameters

image	an image

3.1.2.2 getImageById()

```
Image Database::getImageById (
            const int id )
```

Get an image with a specific id.

Parameters

id	a image's identifier

Returns

the image with the given id

3.1.2.3 getImages()

```
std::vector< Image > Database::getImages ( ) const
```

Get all the images stored in the database.

Returns

a vector with all the images

3.1.2.4 load()

```
void Database::load (
            const std::string & directoryName )
```

Load database using files sotred in a specified directory.

Parameters

directoryName	the path to thie directory containing the descriptor files

3.1.2.5 setImages()

```
void Database::setImages (
            const std::vector< Image > & images )
```

Define the images in database.

Parameters

images	a vector of images

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

- database/Database.h
- database/Database.cpp

3.2 Image Class Reference

Public Member Functions

- Image ()
  Default constructor.
- Image (const ImageDescriptor &)
  Creates an Image which stores this matrix and the descriptor.
- cv::Mat getImage () const
  Get the matrix storing the image.
- ImageDescriptor getImageDescriptor () const
  Get the image's descriptor.
- void setImage (const cv::Mat &)
  Set the matrix of the image.
- void setImageDescriptor (const ImageDescriptor &)
  Set the descriptor for this image.
- void **loadImage** ()
  Create the matrix storing the image.
- void save (const std::string &) const
  Save the image into a file.
- void **show** () const
  Show the image using cv::imshow.
- void setColorModel (const cv::ColorConversionCodes code)
  Change the image's color model.
- std::vector< std::vector< int > > histogram ()
  Create the histogram of each color channel.
- std::vector< int > cumulativeHistogram (const std::vector< int > &)
  Calculate the cumulative histogram of the given histogram.
- void **normalizeHistogram** ()
  Normalize the histogram of the image.
- void equalizeHistogram (const int, const int)
  Equalize the histogram of the image.
- void convolution (cv::Mat &, const int)

Apply by convolution a kernel to the image.

- cv::Mat gradient ()

  Calculate the gradient of the grayscale image.

- void borderEnhancement ()

  Enhance the borders of the objects in the grayscale image.

- void applyMedianFilter (const int)

  Apply median filter to image.

- void applyErosionFilter (const int, const int)

  Erode the image.

- void applyDilationFilter (const int, const int)

  Dilate the image.

- void colorSegmentation (const int, const int, const int, const int, const int, const int)

  Segment the image by chosing upper and lower bound parameters for each color channel.

- void saltpepperNoise (const int)

  Add salt pepper noise to the image.

- void gaussianNoise (double)

  Add gaussian noise to the image.

- void detectLines (const int, const int, const int)

3.2.1 Constructor & Destructor Documentation

3.2.1.1 Image() [1/2]

```
Image::Image ( )
```

Default constructor.

The descriptor of the image

3.2.1.2 Image() [2/2]

```
Image::Image (
            const ImageDescriptor & descriptor )
```

Creates an Image which stores this matrix and the descriptor.

Parameters

descriptor	a descriptor to give to the image

3.2.2 Member Function Documentation

3.2.2.1 applyDilationFilter()

```
void Image::applyDilationFilter (
            const int iterations = 1,
            const int kernelSize = 3 )
```

Dilate the image.

Parameters

iterations	maximum number of iterations
kernelSize	the size of the matrix that will be applied to the image as kernel

3.2.2.2 applyErosionFilter()

```
void Image::applyErosionFilter (
            const int iterations = 1,
            const int kernelSize = 3 )
```

Erode the image.

Parameters

iterations	maximum number of iterations
kernelSize	the size of the matrix that will be applied to the image as kernel

3.2.2.3 applyMedianFilter()

```
void Image::applyMedianFilter (
            const int kernelSize = 3 )
```

Apply median filter to image.

Parameters

kernelSize	the size of the matrix that will be applied to the image as kernel

3.2.2.4 borderEnhancement()

```
void Image::borderEnhancement ( )
```

Enhance the borders of the objects in the grayscale image.

Note

The image is calculated by the following formula: enhanced = grayscale + (0.8 ∗ gradient)

3.2.2.5 colorSegmentation()

```
void Image::colorSegmentation (
            const int lowR,
            const int highR,
            const int lowG,
            const int highG,
            const int lowB,
            const int highB )
```

Segment the image by chosing upper and lower bound parameters for each color channel.

Parameters

lowR	the lower bound for the red channel
highR	the higher bound for the red channel
lowG	the lower bound for the green channel
highG	the higher bound for the green channel
lowB	the lower bound for the blue channel
highB	the higher bound for the blue channel

3.2.2.6 convolution()

```
void Image::convolution (
            cv::Mat & kernel,
            const int mode )
```

Apply by convolution a kernel to the image.

Parameters

kernel	a matrix that will be applied to the image
mode	an integer (1: use opencv's filter2D function)

3.2.2.7 cumulativeHistogram()

```
std::vector< int > Image::cumulativeHistogram (
            const std::vector< int > & histogram )
```

Calculate the cumulative histogram of the given histogram.

Parameters

histogram	a histogram

Returns

the cumulative histogram

3.2.2.8 detectLines()

```
void Image::detectLines (
            const int threshold,
            const int minLineLength,
            const int maxLineGap )
```

Parameters

threshold	the threshold
minLineLength	the minimum line length
maxLineGap	the maximum line gap

3.2.2.9 equalizeHistogram()

```
void Image::equalizeHistogram (
            const int min,
            const int max )
```

Equalize the histogram of the image.

Parameters

min	minimum imgae's dynamic value
max	maximum image's dynamic value

3.2.2.10 gaussianNoise()

```
void Image::gaussianNoise (
            double variance )
```

Add gaussian noise to the image.

Parameters

variance	the gaussian noise variance

3.2.2.11 getImage()

```
cv::Mat Image::getImage ( ) const
```

Get the matrix storing the image.

Returns

the matrix storing the image

3.2.2.12 getImageDescriptor()

```
ImageDescriptor Image::getImageDescriptor ( ) const
```

Get the image's descriptor.

Returns

the image descriptior

3.2.2.13 gradient()

```
cv::Mat Image::gradient ( )
```

Calculate the gradient of the grayscale image.

Returns

the gradient image

3.2.2.14 histogram()

```
std::vector< std::vector< int > > Image::histogram ( )
```

Create the histogram of each color channel.

Returns

a vector for a grayscale image

Note

The vector contains every histogram for each color chanel

3.2.2.15 saltpepperNoise()

```
void Image::saltpepperNoise (
            const int percentage )
```

Add salt pepper noise to the image.

Parameters

percentage	the noise percentage

3.2.2.16 save()

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

Save the image into a file.

Parameters

path	the path to save the image

3.2.2.17 setColorModel()

```
void Image::setColorModel (
            const cv::ColorConversionCodes code )
```

Change the image's color model.

Parameters

code	a opencv color conversion code

3.2.2.18 setImage()

```
void Image::setImage (
            const cv::Mat & image )
```

Set the matrix of the image.

Parameters

image	a matrix storing an image

3.2.2.19 setImageDescriptor()

```
void Image::setImageDescriptor (
            const ImageDescriptor & descriptor )
```

Set the descriptor for this image.

Parameters

descriptor	a descriptor for the image

Note

This will reload the image due to the fact that the descriptor stores the path to the image

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

- image/Image.h
- image/Image.cpp

3.3 ImageDescriptor Class Reference

Public Member Functions

- ImageDescriptor (const int, const std::string &, const std::string &, const std::string &, const std::string &, const std::string &, const int)
- ImageDescriptor (const std::string &, const std::string &, const std::string &, const std::string &, const std::string &, const int)
- int **getId** () const
- std::string **getPath** () const
- std::string **getTitle** () const
- std::string **getSource** () const
- std::string **getAuthor** () const
- std::string **getAccessLevel** () const
- int **getWeight** () const
- void setId (const int)
- void setPath (const std::string &)
- void setTitle (const std::string &)
- void setSource (const std::string &)
- void setAuthor (const std::string &)
- void setAccessLevel (const std::string &)
- void setWeight (const int)
- void save (const std::string &) const

3.3.1 Constructor & Destructor Documentation

3.3.1.1 ImageDescriptor() [1/2]

```
ImageDescriptor::ImageDescriptor (
            const int id,
            const std::string & path,
            const std::string & title,
            const std::string & source,
            const std::string & author,
            const std::string & access,
            const int weight )
```

Parameters

id	an identifier
path	the image's path
title	the image's title
source	this image's source
author	this image's author
access	the image's access level (public or private)
weight	this image's weight

3.3.1.2 ImageDescriptor() [2/2]

```
ImageDescriptor::ImageDescriptor (
            const std::string & path,
            const std::string & title,
            const std::string & source,
            const std::string & author,
            const std::string & access,
            const int weight )
```

Parameters

path	the image's path
title	the image's title
source	this image's source
author	this image's author
access	the image's access level (public or private)
weight	this image's weight

3.3.2 Member Function Documentation

3.3.2.1 save()

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

Parameters

path	the path to where the image will be saved

3.3.2.2 setAccessLevel()

```
void ImageDescriptor::setAccessLevel (
            const std::string & access )
```

Parameters

access	the image's access level

3.3.2.3 setAuthor()

```
void ImageDescriptor::setAuthor (
            const std::string & author )
```

Parameters

author	the image's author

3.3.2.4 setId()

```
void ImageDescriptor::setId (
            const int id )
```

Parameters

id	an identifier

3.3.2.5 setPath()

```
void ImageDescriptor::setPath (
            const std::string & path )
```

Parameters

path	a path to the image's location

3.3.2.6 setSource()

```
void ImageDescriptor::setSource (
            const std::string & source )
```

Parameters

source	the image's source

3.3.2.7 setTitle()

```
void ImageDescriptor::setTitle (
            const std::string & title )
```

Parameters

title	a title for the image

3.3.2.8 setWeight()

```
void ImageDescriptor::setWeight (
            const int weight )
```

Parameters

weight	the image's weight

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

- descriptor/ImageDescriptor.h
- descriptor/ImageDescriptor.cpp

3.4 User Class Reference

Public Member Functions

- User ()

- **User** (const std::string &, const std::string &)
- std::string getUsername () const

  Get the users's username.
- std::string getPassword () const

  Get the users's password.
- bool isAdmin () const

  Check if the users is an administrator.
- void setUsername (const std::string &)

  Set the users's username.
- void setPassword (const std::string &)

  Set the users's password.
- void setAdmin (const bool)

  Set administator rights.
- bool verifyLogin (const std::string &)

  Check if the users can login to the app.

3.4.1 Constructor & Destructor Documentation

3.4.1.1 User()

```
User::User ( )
```

The users's role (by default he is not an administrator)

3.4.2 Member Function Documentation

3.4.2.1 getPassword()

```
std::string User::getPassword ( ) const
```

Get the users's password.

Returns

the users's password

3.4.2.2 getUsername()

```
std::string User::getUsername ( ) const
```

Get the users's username.

Returns

the users's username

3.4.2.3 isAdmin()

```
bool User::isAdmin ( ) const
```

Check if the users is an administrator.

Returns

whether the users is an administrator or not

3.4.2.4 setAdmin()

```
void User::setAdmin (
            const bool isAdmin )
```

Set administator rights.

Parameters

isAdmin	whether or not the iuser is going to be an administrator

3.4.2.5 setPassword()

```
void User::setPassword (
            const std::string & password )
```

Set the users's password.

Parameters

password	a password

3.4.2.6 setUsername()

```
void User::setUsername (
            const std::string & username )
```

Set the users's username.

Parameters

username	a username

3.4.2.7 verifyLogin()

```
bool User::verifyLogin (
            const std::string & path )
```

Check if the users can login to the app.

Parameters

path	The directory where users files are stored

Returns

whether the users can login or not

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

- user/User.h
- user/User.cpp

Chapter 4

File Documentation

4.1 Database.h

```
1 #ifndef DATABSE_H
2 #define DATABSE_H
3
4 #include <vector>
5
6 #include "../image/Image.h"
7
8 class Database {
9     private:
10         std::vector<Image> _images;
12     public:
16         Database();
17
21         void load(const std::string &);
22
26         std::vector<Image> getImages() const;
27
31         Image getImageById(const int);
32
36         void setImages(const std::vector<Image> &);
37
41         void addImage(const Image &);
42 };
43
44 std::ostream &operator«(std::ostream &, const Database &);
45
46 #endif //DATABSE_H
```

4.2 ImageDescriptor.h

```
1 #ifndef IMAGEDESCRIPTOR_H
2 #define IMAGEDESCRIPTOR_H
3
4 #include <iostream>
5 #include <string>
6
7 class ImageDescriptor {
8     private:
9     int _id, _weight;
10     std::string _path, _title, _source, _author, _access;
11
12     public:
13         // Constructors
14         ImageDescriptor();
15         ImageDescriptor(const int, const std::string &, const std::string &, const std::string &, const
      std::string &, const std::string &, const int);
16         ImageDescriptor(const std::string &, const std::string &, const std::string &, const std::string
      &, const std::string &, const int);
17
18         // Getters
19         int getId() const;
20         std::string getPath() const;
21         std::string getTitle() const;
22         std::string getSource() const;
```

```
23          std::string getAuthor() const;
24          std::string getAccessLevel() const;
25          int getWeight() const;
26
27          // Settters
28          void setId(const int);
29          void setPath(const std::string &);
30          void setTitle(const std::string &);
31          void setSource(const std::string &);
32          void setAuthor(const std::string &);
33          void setAccessLevel(const std::string &);
34          void setWeight(const int);
35          void save(const std::string&) const;
36 };
37
38 std::ostream &operator«(std::ostream &, const ImageDescriptor &);
39
40 #endif //IMAGEDESCRIPTOR_H
```

4.3 Image.h

```
1 #ifndef IMAGE_H
2 #define IMAGE_H
3
4 #include <opencv2/opencv.hpp>
5
6 #include "../descriptor/ImageDescriptor.h"
7
8 class Image {
9     private:
10          cv::Mat _image;
11          ImageDescriptor _descriptor;
13     public:
17          Image();
18
23          Image(const ImageDescriptor &);
24
28          cv::Mat getImage() const;
29
33          ImageDescriptor getImageDescriptor() const;
34
38          void setImage(const cv::Mat &);
39
43          void setImageDescriptor(const ImageDescriptor &);
44
48          void loadImage();
49
53          void save(const std::string &) const;
54
58          void show() const;
59
63          void setColorModel(const cv::ColorConversionCodes code);
64
68          std::vector<std::vector<int» histogram();
69
73          std::vector<int> cumulativeHistogram(const std::vector<int> &);
74
78          void normalizeHistogram();
79
83          void equalizeHistogram(const int, const int);
84
88          void convolution(cv::Mat &, const int);
89
93          cv::Mat gradient();
94
98          void borderEnhancement();
99
103          void applyMedianFilter(const int);
104
108          void applyErosionFilter(const int, const int);
109
113          void applyDilationFilter(const int, const int);
114
118          void colorSegmentation(const int, const int, const int, const int, const int, const int);
119
123          void saltpepperNoise(const int);
124
128          void gaussianNoise(double);
129
130          /*
131 * \brief Detect lines into the image
132 */
133          void detectLines(const int, const int, const int);
```

```
134 };
135
136 #endif //IMAGE_H
```

4.4 User.h

```
1 #ifndef USER_H
2 #define USER_H
3
4 #include <string>
5
6 class User {
7     private:
8         std::string _username, _password;
9         bool _isAdmin = false;
11     public:
12         User();
13         User(const std::string &, const std::string &);
14
18         std::string getUsername() const;
19
23         std::string getPassword() const;
24
28         bool isAdmin() const;
29
33         void setUsername(const std::string &);
34
38         void setPassword(const std::string &);
39
43         void setAdmin(const bool);
44
48         bool verifyLogin(const std::string &);
49 };
50
51 #endif //USER_H
```

Index