FSP_ImageDeconvolution

0.1

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 AppLogic Namespace Reference

Classes

- class Filter

  Allows to create filters based on training sets of images.
- struct FilterInfo

  struct to hold the the information of the calculated filter
- class FilterLogic

  API to the Filter class.
- class ImageCleaningLogic

  API to the ImageDeconvolution class.
- class ImageDeconvolution

  Allows to apply existing filter to images.
- class ImageLoader

  Allows to load images and filter information as well as import and export filters.
- class VecImage

  Class used for image manipulation.

Enumerations

- enum noise_type_t {
  GAUSSIAN, SALT_PEPPER, UNIFORM, POISSON,
  RICIAN }

  enum with the different type of noises that can be applied to a `VecImage`
- enum calc_method_t { RCIMA_METHOD, FAST_RCIMA_METHOD }

  enum with the calculation methods for the filter

Functions

- std::ostream & operator<< (std::ostream &stream, FilterInfo const &data)

  outputs filter information to a stream. Used to save FilterInfo to a file.
- std::istream & operator>> (std::istream &stream, FilterInfo &data)

  operator to read filter information from a stream. used to read FilterInfo from file

5.1.1 Enumeration Type Documentation

5.1.1.1 calc_method_t

```
enum AppLogic::calc_method_t
```

enum with the calculation methods for the filter

Enumerator

RCIMA_METHOD	
FAST_RCIMA_METHOD	

5.1.1.2 noise_type_t

```
enum AppLogic::noise_type_t
```

enum with the different type of noises that can be applied to a VecImage

Enumerator

GAUSSIAN	
SALT_PEPPER	
UNIFORM	
POISSON	
RICIAN	

5.1.2 Function Documentation

5.1.2.1 operator<<()

```
std::ostream& AppLogic::operator<< (
            std::ostream & stream,
            FilterInfo const & data )  [inline]
```

outputs filter information to a stream. Used to save FilterInfo to a file.

Parameters

stream	where the filter information is written
data	filter information to write to stream

Returns

std::ostream& returns stream with the filter information written

5.1.2.2 operator>>()

```
std::istream& AppLogic::operator>> (
            std::istream & stream,
            FilterInfo & data )  [inline]
```

operator to read filter information from a stream. used to read FilterInfo from file

Parameters

stream	stream to read filter information from
data	FilterInfo struct where the read information is written

Returns

std::istream& returns stream with the information read

5.2 frcima Namespace Reference

Functions

- mat rcima (const vector< vector< double >> &t_data_x, const vector< vector< double >> &t_data_c, const size_t rank)

  calculates rank constrained inverse matrix approximation using RCIMA method
- mat rcima (const mat &X, const mat &C, const size_t rank)

  calculates rank constrained inverse matrix approximation using RCIMA method
- mat fast_rcima (const vector< vector< double >> &t_data_x, const vector< vector< double >> &t_data_c, const size_t rank)

  calculates rank constrained inverse matrix approximation using fast-RCIMA algorithm
- mat fast_rcima (const mat &X, const mat &C, const size_t rank)

  calculates rank constrained inverse matrix approximation using fast-RCIMA algorithm
- void calculate_error (mat &A, const mat &C, const mat &X)

  calculates the error from the calculated filter
- bool low_rank_approx (const mat &A, const size_t r, mat &B, mat &C)

  calculates low rank matrix approximation using the modified bilateral random projections (MBRP) method.
- bool low_rank (const mat &A, const size_t r, mat &B, mat &C)

  calculates low rank matrix using SVD method
- bool generate_training_matrices (const vector< vector< double >> &X, const vector< vector< double >> &C, mat &X_tr, mat &C_tr)

  generates training matrices from vectors
- bool pseudo_inverse_tpm (const mat &A, mat &Ap)

  Calculates pseudo inverse matrix using the Tensor Product Matrix (TPM) method.

Variables

- double _calculated_error

  global variable to hold calculated error

5.2.1 Function Documentation

5.2.1.1 calculate_error()

```
void frcima::calculate_error (
            mat & A,
            const mat & C,
            const mat & X )
```

calculates the error from the calculated filter

Parameters

A	Calculated filter we want to get the error from
C	Training matrix with vectorized noisy images
X	Training matrix with vectorized source images

5.2.1.2 fast_rcima() [1/2]

```
mat frcima::fast_rcima (
            const vector< vector< double >> & t_data_x,
            const vector< vector< double >> & t_data_c,
            const size_t rank )
```

calculates rank constrained inverse matrix approximation using fast-RCIMA algorithm

Parameters

t_data_x	the original source images. The training data is a vector of vectors containing a vectorized form of an image
t_data_c	the images with noise. The training data is a vector of vectors containing a vectorized form of an image
rank	desired rank approximation, should be less than the size of training data

Returns

mat rank constrained inverse matrix approximation

5.2.1.3 fast_rcima() [2/2]

```
mat frcima::fast_rcima (
              const mat & X,
              const mat & C,
              const size_t rank )
```

calculates rank constrained inverse matrix approximation using fast-RCIMA algorithm

Parameters

t_data_x	the original source images. The training data is an arma::mat where each column is a vectorized form of a training image
t_data_c	the images with noise. The training data is an arma::mat where each column is a vectorized form of a training image
rank	desired rank approximation, should be less than the size of training data

Returns

mat rank constrained inverse matrix approximation

5.2.1.4 generate_training_matrices()

```
bool frcima::generate_training_matrices (
              const vector< vector< double >> & X,
              const vector< vector< double >> & C,
              mat & X_tr,
              mat & C_tr )
```

generates training matrices from vectors

Parameters

X	input training vector
C	input training vector
X_tr	ouput training matrix
C_tr	output training matrixñ

Returns

true if successfull generating matrices
false if falure in process

5.2.1.5 low_rank()

```
bool frcima::low_rank (
             const mat & A,
             const size_t r,
             mat & B,
             mat & C )
```

calculates low rank matrix using SVD method

Parameters

A	input matrix to calculate low rank
r	desired rank, must be less than the minimun between the number of columns and rows of A
B	output left bilateral matrix
C	output right bilateral matrix

Returns

true if low rank approximation is calculated
false if there is a problem during calculation

5.2.1.6 low_rank_approx()

```
bool frcima::low_rank_approx (
             const mat & A,
             const size_t r,
             mat & B,
             mat & C )
```

calculates low rank matrix approximation using the modified bilateral random projections (MBRP) method.

Parameters

A	input matrix to calculate low rank approximation
r	desired rank, must be less than the minimun between the number of columns and rows of A
B	output left bilateral matrix
C	output right bilateral matrix

Returns

true if low rank approximation is calculated
false if there is a problem during calculation

5.2.1.7 pseudo_inverse_tpm()

```
bool frcima::pseudo_inverse_tpm (
            const mat & A,
            mat & Ap )
```

Calculates pseudo inverse matrix using the Tensor Product Matrix (TPM) method.

Parameters

A	Matrix to calculate pseudoinverse
Ap	Output calculated pseudoinverse matrix

Returns

true and Ap matrix contains the calculated pseudo inverse matrix
false

5.2.1.8 rcima() [1/2]

```
mat frcima::rcima (
            const vector< vector< double >> & t_data_x,
            const vector< vector< double >> & t_data_c,
            const size_t rank )
```

calculates rank constrained inverse matrix approximation using RCIMA method

Parameters

t_data_x	the original source images. The training data is a vector of vectors containing a vectorized form of an image
t_data_c	the images with a noise. The training data is a vector of vectors containing a vectorized form of an image
rank	desired rank constrain, should be less than the size of training data

Returns

mat rank constrained inverse matrix approximation

5.2.1.9 rcima() [2/2]

```
mat frcima::rcima (
            const mat & X,
            const mat & C,
            const size_t rank )
```

calculates rank constrained inverse matrix approximation using RCIMA method

Parameters

t_data_x	the original source images. The training data is an arma::mat where each column is a vectorized form of a training image
t_data_c	the images with a noise applied. The training data is an arma::mat where each column is a vectorized form of a training image
rank	desired rank constrain, should be less than the size of training data

Returns

mat rank constrained inverse matrix approximation

5.2.2 Variable Documentation

5.2.2.1 _calculated_error

```
double frcima::_calculated_error
```

global variable to hold calculated error

5.3 UI Namespace Reference

Classes

- class FilterWindow

  Window to create filters.
- class ImageDeconvolutionWindow

  Window to filter images.

Variables

- static const char notAllowedChars [ ] = "/\\,^@={}[ ]~!?:&*\"|#%<>$\'();" "
- static const char * notAllowedSubStrings [ ] = {".."}

5.3.1 Variable Documentation

5.3.1.1 notAllowedChars

```
const char UI::notAllowedChars[] = "/\\,^@={}[]~!?:&*\"|#%<>$\"'();`' "  [static]
```

5.3.1.2 notAllowedSubStrings

```
const char* UI::notAllowedSubStrings[] = {".."}  [static]
```

5.4 Ui Namespace Reference

5.5 uihelp Namespace Reference

Enumerations

- enum file_dialog_type_t { FD_IMAGE, FD_ZIP }

  enum with file dialog type

Functions

- static QString allSupportedFormatsString (QStringList mimeTypeFilters)

  constructs string with all supported MIME types
- static void initializeImageFileDialog (QFileDialog &dialog, QFileDialog::AcceptMode acceptMode, file_dialog_type_t file_dialog_type=file_dialog_type_t::FD_IMAGE, QFileDialog::FileMode filemode=QFileDialog::FileMode::DirectoryOnly)

  creates a file dialog window
- static void hideListItems (QListWidget *list)

  hides all items in a QListWidget
- static void filterListItems (QListWidget *list, QString filter_string)

  filters the list contents that contain the filter string
- static QString generateSizeMessage (int rows, int cols)

  generates string with size information

5.5.1 Enumeration Type Documentation

5.5.1.1 file_dialog_type_t

```
enum uihelp::file_dialog_type_t
```

enum with file dialog type

Enumerator

FD_IMAGE	an image file dialog
FD_ZIP	file dialog for zip files

5.5.2 Function Documentation

5.5.2.1 allSupportedFormatsString()

```
static QString uihelp::allSupportedFormatsString (
            QStringList mimeTypeFilters )  [inline], [static]
```

constructs string with all supported MIME types

Parameters

mimeTypeFilters	a list of strings of the MIME types

Returns

QString a string with all valid extensions of the MIME type

5.5.2.2 filterListItems()

```
static void uihelp::filterListItems (
            QListWidget * list,
            QString filter_string )  [inline], [static]
```

filters the list contents that contain the filter string

Parameters

list	list of items to filter
filter_string	contains the string to filter for

5.5.2.3 generateSizeMessage()

```
static QString uihelp::generateSizeMessage (
            int rows,
            int cols )  [inline], [static]
```

generates string with size information

Parameters

rows	amount of rows
cols	amount of columns

Returns

QString formatted size message string

5.5.2.4 hideListItems()

```
static void uihelp::hideListItems (
            QListWidget * list )  [inline], [static]
```

hides all items in a QListWidget

Parameters

list	lis widget to hide items for

5.5.2.5 initializeImageFileDialog()

```
static void uihelp::initializeImageFileDialog (
            QFileDialog & dialog,
            QFileDialog::AcceptMode acceptMode,
            file_dialog_type_t file_dialog_type = file_dialog_type_t::FD_IMAGE,
            QFileDialog::FileMode filemode = QFileDialog::FileMode::DirectoryOnly )  [inline],
[static]
```

creates a file dialog window

Parameters

dialog	pointer to the file dialog window
acceptMode	whether it is to save or load file
file_dialog_type	type of dialog window, either FD_ZIP or FD_IMAGE
filemode	whether it is looking for a file or a directory

Chapter 6

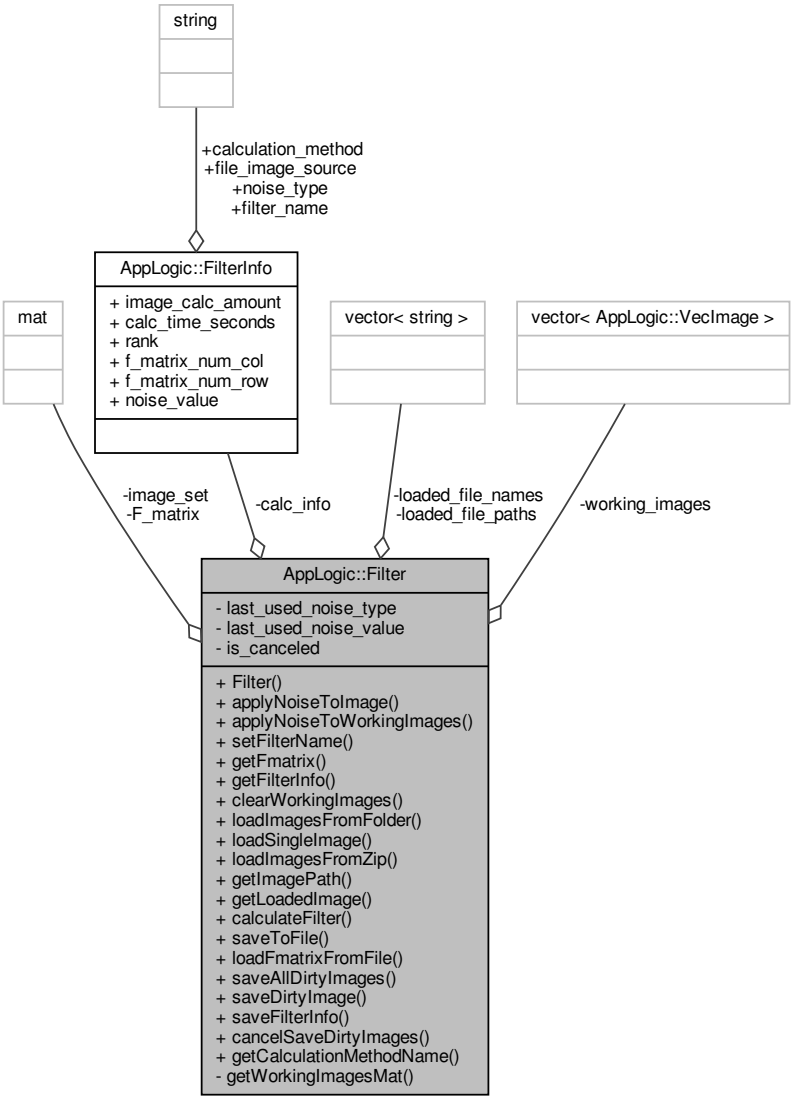
Class Documentation

6.1 AppLogic::Filter Class Reference

Allows to create filters based on training sets of images.

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

Collaboration diagram for AppLogic::Filter:



Public Member Functions

- Filter ()

  Construct a new Filter object.
- VecImage applyNoiseToImage (size_t image_id, size_t noise_value, noise_type_t noise_type=noise_type_t::GAUSSIAN)

  applies selected noise to an image and returns a new image object
- void applyNoiseToWorkingImages (size_t noise_value, noise_type_t noise_type=noise_type_t::GAUSSIAN)
- void setFilterName (string name)

  sets the name of the filter
- mat & getFmatrix ()

  Get the Fmatrix object.
- FilterInfo * getFilterInfo ()

Get the Filter Info object.

- void clearWorkingImages ()

  removes all loaded image information
- vector< string > loadImagesFromFolder (string folder_path)

  loads images from specified full path to a folder
- string loadSingleImage (string folder_path)

  loads a single image and adds it to the working images
- vector< string > loadImagesFromZip (string file_path)

  load images from a zip file
- string getImagePath (size_t index)

  returns the path of the loaded image specified by index
- VecImage * getLoadedImage (const size_t index)

  returns a pointer to the loaded VecImage specified by index
- bool calculateFilter (size_t rank, calc_method_t calc_method)

  calculates the filter
- bool saveToFile (string file_path=ImageLoader::FILTER_SAVE_LOCATION)

  saves filter matrix to file
- bool loadFmatrixFromFile (string file_path)

  loads filter matrix from file
- bool saveAllDirtyImages (string folder_path)

  applies noise to all working images and saves them in the specified path
- bool saveDirtyImage (size_t image_id, string folder_path)

  applies noise to image and saves it in the specified path
- bool saveFilterInfo (string folder_path=ImageLoader::FILTER_SAVE_LOCATION)

  saves filter calculation information in the app's data directory
- void cancelSaveDirtyImages ()

  cancels the process of saving images with noise applied

Static Public Member Functions

- static string getCalculationMethodName (calc_method_t calc_method)

  returns a string name for the calculation method

Private Member Functions

- mat getWorkingImagesMat ()

  generates training matrix out of the loaded images

Private Attributes

- FilterInfo calc_info

  filter calculation information
- vector< VecImage > working_images

  list of loaded images
- mat image_set

  training matrix of loaded images
- vector< string > loaded_file_paths

  paths to the loaded image files
- vector< string > loaded_file_names

names of the loaded images

- noise_type_t last_used_noise_type

  last noise type used

- size_t last_used_noise_value

  last noise value used

- bool is_canceled

  indicates whether filter calculation has been canceled

- mat F_matrix

  created filter matrix

6.1.1 Detailed Description

Allows to create filters based on training sets of images.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Filter()

```
AppLogic::Filter::Filter ( )
```

Construct a new Filter object.

6.1.3 Member Function Documentation

6.1.3.1 applyNoiseToImage()

```
VecImage AppLogic::Filter::applyNoiseToImage (
            size_t image_id,
            size_t noise_value,
            noise_type_t noise_type = noise_type_t::GAUSSIAN )
```

applies selected noise to an image and returns a new image object

Parameters

image_id	the index of the loaded image to apply noise to
noise_value	percentage value (0-99) of amount of noise to apply
noise_type	type of noise to apply to the image

Returns

VecImage new VecImage with the applied noise

6.1.3.2 applyNoiseToWorkingImages()

```
void AppLogic::Filter::applyNoiseToWorkingImages (
            size_t noise_value,
            noise_type_t noise_type = noise_type_t::GAUSSIAN )
```

6.1.3.3 calculateFilter()

```
bool AppLogic::Filter::calculateFilter (
            size_t rank,
            calc_method_t calc_method )
```

calculates the filter

Calculates the filter using the last noise type and value applied, rank, and selected calculation method. Sets the calculation information in `calc_info`

Parameters

rank	the rank to use for filter calculation. Must be a value between 1 and the minimun between the amount of images used and the size of the vectorized image.
calc_method	calculation method to use. Either RCIMA or fast-RCIMA

Returns

true if calculation is successful
false if an error is encountered during calculation or calculation is canceled

6.1.3.4 cancelSaveDirtyImages()

```
void AppLogic::Filter::cancelSaveDirtyImages ( )
```

cancels the process of saving images with noise applied

6.1.3.5 clearWorkingImages()

```
void AppLogic::Filter::clearWorkingImages ( )  [inline]
```

removes all loaded image information

6.1.3.6 getCalculationMethodName()

```
string AppLogic::Filter::getCalculationMethodName (
            calc_method_t calc_method ) [static]
```

returns a string name for the calculation method

Parameters

calc_method	calculation method

Returns

string name of the calculation method

6.1.3.7 getFilterInfo()

```
FilterInfo* AppLogic::Filter::getFilterInfo ( ) [inline]
```

Get the Filter Info object.

Returns

FilterInfo* filter calculation information

6.1.3.8 getFmatrix()

```
mat& AppLogic::Filter::getFmatrix ( ) [inline]
```

Get the Fmatrix object.

Returns

mat& filter matrix

6.1.3.9 getImagePath()

```
string AppLogic::Filter::getImagePath (
            size_t index )
```

returns the path of the loaded image specified by index

Parameters

index	the index of the loaded image to get the path for

Returns

string path of the loaded image in the filesystem

6.1.3.10 getLoadedImage()

```
VecImage * AppLogic::Filter::getLoadedImage (
            const size_t index )
```

returns a pointer to the loaded VecImage specified by index

Parameters

index	the index of the loaded image to get

Returns

VecImage* pointer to the VecImage object of the loaded image

6.1.3.11 getWorkingImagesMat()

```
mat AppLogic::Filter::getWorkingImagesMat ( )  [private]
```

generates training matrix out of the loaded images

Returns

mat training matrix where each column is a vetorized form of each loaded image

6.1.3.12 loadFmatrixFromFile()

```
bool AppLogic::Filter::loadFmatrixFromFile (
            string file_path )
```

loads filter matrix from file

Parameters

file_path	full path of the file where the matrix is stored

Returns

true if matrix is loaded successfully
false if there is an error loading the matrix

6.1.3.13 loadImagesFromFolder()

```
vector< string > AppLogic::Filter::loadImagesFromFolder (
            string folder_path )
```

loads images from specified full path to a folder

This function will look for the first valid image in the folder and will load all other valid images of the same resolution as that first loaded image, other images are ignored.

Parameters

folder_path	full path to the directory containing the images

Returns

vector<string> list of names of the images that were loaded

6.1.3.14 loadImagesFromZip()

```
vector< string > AppLogic::Filter::loadImagesFromZip (
            string file_path )
```

load images from a zip file

This function will go through all files in a zip file. It will look for the first valid image in the folder and will load all other valid images of the same resolution as the first loaded image, other images are ignored.

Parameters

file_path	full path of a valid zip file

Returns

vector<string> list of names of the images that were loaded

6.1.3.15 loadSingleImage()

```
string AppLogic::Filter::loadSingleImage (
            string folder_path )
```

loads a single image and adds it to the working images

Parameters

file_path	path of the image file

Returns

string name of the loaded image

6.1.3.16 saveAllDirtyImages()

```
bool AppLogic::Filter::saveAllDirtyImages (
            string folder_path )
```

applies noise to all working images and saves them in the specified path

Applies noise to all loaded images using the last noise type and value applied and saves them to the folder path provided as PNG (.png) images. This operation does not modify the loaded images.

Parameters

folder_path	path where the images will be saved

Returns

true if images are saved correctly
false if an eror is encountered or saving process is canceled

6.1.3.17 saveDirtyImage()

```
bool AppLogic::Filter::saveDirtyImage (
            size_t image_id,
            string folder_path )
```

applies noise to image and saves it in the specified path

Applies noise to the loaded image specified by the index image_id, using the last noise type and value applied and saves it to the folder path provided as a PNG (.png) image. This operation does not modify the loaded image.

Parameters

image_id	index of the image to apply noise to and save
folder_path	name of the file to save the dirty image

Returns

true if image is saved corectly
false if there is an error saving the image

6.1.3.18 saveFilterInfo()

```
bool AppLogic::Filter::saveFilterInfo (
            string folder_path = ImageLoader::FILTER_SAVE_LOCATION )
```

saves filter calculation information in the app's data directory

Parameters

folder_path	path where the filter_info is stored

Returns

true if filter information is saved correctly
false if there is an error saving the filter information

6.1.3.19 saveToFile()

```
bool AppLogic::Filter::saveToFile (
            string file_path = ImageLoader::FILTER_SAVE_LOCATION )
```

saves filter matrix to file

Parameters

file_path	full path of the file where the matrix will be saved

Returns

true if matrix is saved successfully
false if there is an error saving the matrix

6.1.3.20 setFilterName()

```
void AppLogic::Filter::setFilterName (
            string name )  [inline]
```

sets the name of the filter

Parameters

name	name of the filter

6.1.4 Member Data Documentation

6.1.4.1 calc_info

```
FilterInfo AppLogic::Filter::calc_info  [private]
```

filter calculation information

6.1.4.2 F_matrix

```
mat AppLogic::Filter::F_matrix  [private]
```

created filter matrix

6.1.4.3 image_set

```
mat AppLogic::Filter::image_set  [private]
```

training matrix of loaded images

6.1.4.4 is_canceled

```
bool AppLogic::Filter::is_canceled  [private]
```

indicates whether filter calculation has been canceled

6.1.4.5 last_used_noise_type

```
noise_type_t AppLogic::Filter::last_used_noise_type  [private]
```

last noise type used

6.1.4.6 last_used_noise_value

```
size_t AppLogic::Filter::last_used_noise_value  [private]
```

last noise value used

6.1.4.7 loaded_file_names

```
vector<string> AppLogic::Filter::loaded_file_names  [private]
```

names of the loaded images

6.1.4.8 loaded_file_paths

```
vector<string> AppLogic::Filter::loaded_file_paths  [private]
```

paths to the loaded image files

6.1.4.9 working_images

```
vector<VecImage> AppLogic::Filter::working_images  [private]
```

list of loaded images

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

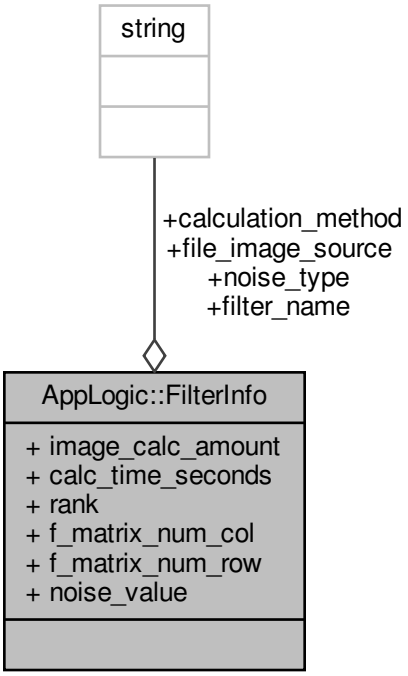
- Filter.hpp
- Filter.cpp

6.2 AppLogic::FilterInfo Struct Reference

struct to hold the the information of the calculated filter

`#include <FilterInfo.hpp>`

Collaboration diagram for AppLogic::FilterInfo:



Public Attributes

- std::string filter_name
  name of the filter
- std::string file_image_source
  name of the directory of the source images
- int image_calc_amount
  amount of images used in the calculation
- float calc_time_seconds
  amount of time taken to calculate the filter
- int rank
  rank used in the calculation
- int f_matrix_num_col
  number of columns of the filter matrix
- int f_matrix_num_row
  number of rows of the filter matrix

- std::string noise_type

  noise type used in the calculation
- double noise_value

  noise value used (this is a percentage from 0 to 99)
- std::string calculation_method

  method used to calculate the filter (either RCIMA of Fast-RCIMA)

6.2.1 Detailed Description

struct to hold the the information of the calculated filter

6.2.2 Member Data Documentation

6.2.2.1 calc_time_seconds

```
float AppLogic::FilterInfo::calc_time_seconds
```

amount of time taken to calculate the filter

6.2.2.2 calculation_method

```
std::string AppLogic::FilterInfo::calculation_method
```

method used to calculate the filter (either RCIMA of Fast-RCIMA)

6.2.2.3 f_matrix_num_col

```
int AppLogic::FilterInfo::f_matrix_num_col
```

number of columns of the filter matrix

6.2.2.4 f_matrix_num_row

```
int AppLogic::FilterInfo::f_matrix_num_row
```

number of rows of the filter matrix

6.2.2.5 file_image_source

```
std::string AppLogic::FilterInfo::file_image_source
```

name of the directory of the source images

6.2.2.6 filter_name

```
std::string AppLogic::FilterInfo::filter_name
```

name of the filter

6.2.2.7 image_calc_amount

```
int AppLogic::FilterInfo::image_calc_amount
```

amount of images used in the calculation

6.2.2.8 noise_type

```
std::string AppLogic::FilterInfo::noise_type
```

noise type used in the calculation

6.2.2.9 noise_value

```
double AppLogic::FilterInfo::noise_value
```

noise value used (this is a percentage from 0 to 99)

6.2.2.10 rank

```
int AppLogic::FilterInfo::rank
```

rank used in the calculation

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

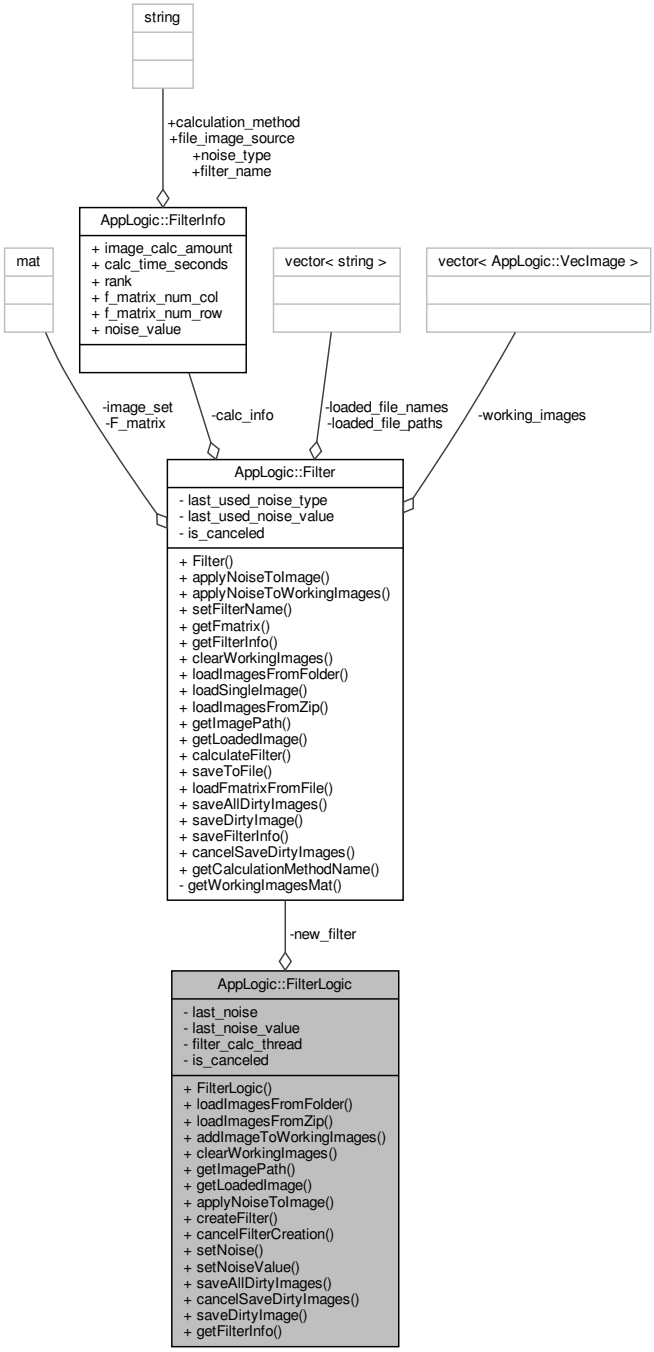
- FilterInfo.hpp

6.3 AppLogic::FilterLogic Class Reference

API to the Filter class.

`#include <FilterLogic.hpp>`

Collaboration diagram for AppLogic::FilterLogic:



Public Member Functions

- FilterLogic ()

  Construct a new Filter Logic object.
- vector< string > loadImagesFromFolder (string folder_path)

  loads images from specified full path to a folder
- vector< string > loadImagesFromZip (string file_path)

  load images from a zip file
- string addImageToWorkingImages (string file_path)

  loads one image and adds it the the list of working images
- void clearWorkingImages ()

  deletes all loaded images
- string getImagePath (const size_t index)

  returns the path of the loaded image specified by index
- VecImage * getLoadedImage (const size_t index)

  returns a pointer to the loaded VecImage specified by index
- VecImage applyNoiseToImage (int image_id, noise_type_t noise, double noise_value)

  applies selected noise to an image and returns a new image object
- bool createFilter (string name, int rank, calc_method_t calc_method)

  calculates the filter
- void cancelFilterCreation ()

  cancels the filter calculation
- void setNoise (noise_type_t n_type)

  sets the value of the last used noise type
- void setNoiseValue (double val)

  sets the last noise value used
- bool saveAllDirtyImages (string folder_path)

  applies noise to all working images and saves them in the specified path
- void cancelSaveDirtyImages ()

  cancels the process of saving images with noise applied
- bool saveDirtyImage (size_t image_id, string folder_path)

  applies noise to image and saves it in the specified path
- FilterInfo * getFilterInfo ()

  returns pointer to the `FilterInfo` object fo current filter.

Private Attributes

- Filter new_filter

  Filter object to use.
- noise_type_t last_noise

  last noise type used
- double last_noise_value

  last noise value used
- pthread_t filter_calc_thread

  thread for calculating the filter
- bool is_canceled = false

  indicates whether filter calculation has been canceled

6.3.1 Detailed Description

API to the Filter class.

Used to provide the functionality of loading images and creating filters out of the loaded images. Allows users to apply noise to images and save the dirty images.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 FilterLogic()

```
AppLogic::FilterLogic::FilterLogic ( )
```

Construct a new Filter Logic object.

6.3.3 Member Function Documentation

6.3.3.1 addImageToWorkingImages()

```
string AppLogic::FilterLogic::addImageToWorkingImages (
            string file_path )
```

loads one image and adds it the the list of working images

Parameters

file_path	full path of a valid image

Returns

string returns the name of the loaded image

6.3.3.2 applyNoiseToImage()

```
VecImage AppLogic::FilterLogic::applyNoiseToImage (
            int image_id,
            noise_type_t noise,
            double noise_value )
```

applies selected noise to an image and returns a new image object

Parameters

image_id	the index of the loaded image to apply noise to
noise	type of noise to apply to the image
noise_value	percentage value (0-99) of amount of noise to apply

Returns

VecImage new VecImage with the applied noise

6.3.3.3 cancelFilterCreation()

```
void AppLogic::FilterLogic::cancelFilterCreation ( )
```

cancels the filter calculation

6.3.3.4 cancelSaveDirtyImages()

```
void AppLogic::FilterLogic::cancelSaveDirtyImages ( )
```

cancels the process of saving images with noise applied

6.3.3.5 clearWorkingImages()

```
void AppLogic::FilterLogic::clearWorkingImages ( )
```

deletes all loaded images

6.3.3.6 createFilter()

```
bool AppLogic::FilterLogic::createFilter (
            string name,
            int rank,
            calc_method_t calc_method )
```

calculates the filter

Calculates the filter using the last noise type and value applied and the selected name, rank, and calculation method

Parameters

name	name of the filter. A file is created with the filter name, no validation is done on the name
rank	the rank to use for filter calculation. Must be a value between 1 and the minimun between the amount of images used and the size of the vectorized image.
calc_method	calculation method to use. Either RCIMA or fast-RCIMA

Returns

true if calculation is successful
false if an error is encountered during calculation or calculation is canceled

6.3.3.7 getFilterInfo()

```
FilterInfo * AppLogic::FilterLogic::getFilterInfo ( )
```

returns pointer to the `FilterInfo` object fo current filter.

Returns

FilterInfo∗ Object containing the calculation information for the curent filter

6.3.3.8 getImagePath()

```
string AppLogic::FilterLogic::getImagePath (
            const size_t index )
```

returns the path of the loaded image specified by index

Parameters

index	the index of the loaded image to get the path for

Returns

string path of the loaded image in the filesystem

6.3.3.9 getLoadedImage()

```
VecImage * AppLogic::FilterLogic::getLoadedImage (
            const size_t index )
```

returns a pointer to the loaded VecImage specified by index

Parameters

index	the index of the loaded image to get

Returns

VecImage* pointer to the VecImage object of the loaded image

6.3.3.10 loadImagesFromFolder()

```
vector< string > AppLogic::FilterLogic::loadImagesFromFolder (
            string folder_path )
```

loads images from specified full path to a folder

This function will look for the first valid image in the folder and will load all other valid images of the same resolution as that first loaded image, other images are ignored.

Parameters

folder_path	full path to the directory containing the images

Returns

vector<string> list of names of the images that were loaded

6.3.3.11 loadImagesFromZip()

```
vector< string > AppLogic::FilterLogic::loadImagesFromZip (
            string file_path )
```

load images from a zip file

This function will go through all files in a zip file. It will look for the first valid image in the folder and will load all other valid images of the same resolution as the first loaded image, other images are ignored.

Parameters

file_path	full path of a valid zip file

Returns

vector<string> list of names of the images that were loaded

6.3.3.12 saveAllDirtyImages()

```
bool AppLogic::FilterLogic::saveAllDirtyImages (
            string folder_path )
```

applies noise to all working images and saves them in the specified path

Applies noise to all loaded images using the last noise type and value applied and saves them to the folder path provided as PNG (.png) images. This operation does not modify the loaded images.

Parameters

folder_path	path where the images will be saved

Returns

true if images are saved correctly
false if an eror is encountered or saving process is canceled

6.3.3.13 saveDirtyImage()

```
bool AppLogic::FilterLogic::saveDirtyImage (
            size_t image_id,
            string folder_path )
```

applies noise to image and saves it in the specified path

Applies noise to the loaded image specified by the index image_id, using the last noise type and value applied and saves it to the folder path provided as a PNG (.png) image. This operation does not modify the loaded image.

Parameters

image_id	index of the image to apply noise to and save
folder_path	name of the file to save the dirty image

Returns

true if image is saved corectly
false if there is an error saving the image

6.3.3.14 setNoise()

```
void AppLogic::FilterLogic::setNoise (
            noise_type_t n_type )  [inline]
```

sets the value of the last used noise type

Parameters

n_type	type of noise

6.3.3.15 setNoiseValue()

```
void AppLogic::FilterLogic::setNoiseValue (
            double val )  [inline]
```

sets the last noise value used

Parameters

val	value in range [0,99] indicating the percentage of noise

6.3.4 Member Data Documentation

6.3.4.1 filter_calc_thread

```
pthread_t AppLogic::FilterLogic::filter_calc_thread  [private]
```

thread for calculating the filter

6.3.4.2 is_canceled

```
bool AppLogic::FilterLogic::is_canceled = false  [private]
```

indicates whether filter calculation has been canceled

6.3.4.3 last_noise

```
noise_type_t AppLogic::FilterLogic::last_noise  [private]
```

last noise type used

6.3.4.4 last_noise_value

```
double AppLogic::FilterLogic::last_noise_value  [private]
```

last noise value used

6.3.4.5 new_filter

```
Filter AppLogic::FilterLogic::new_filter  [private]
```

Filter object to use.

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

- FilterLogic.hpp
- FilterLogic.cpp

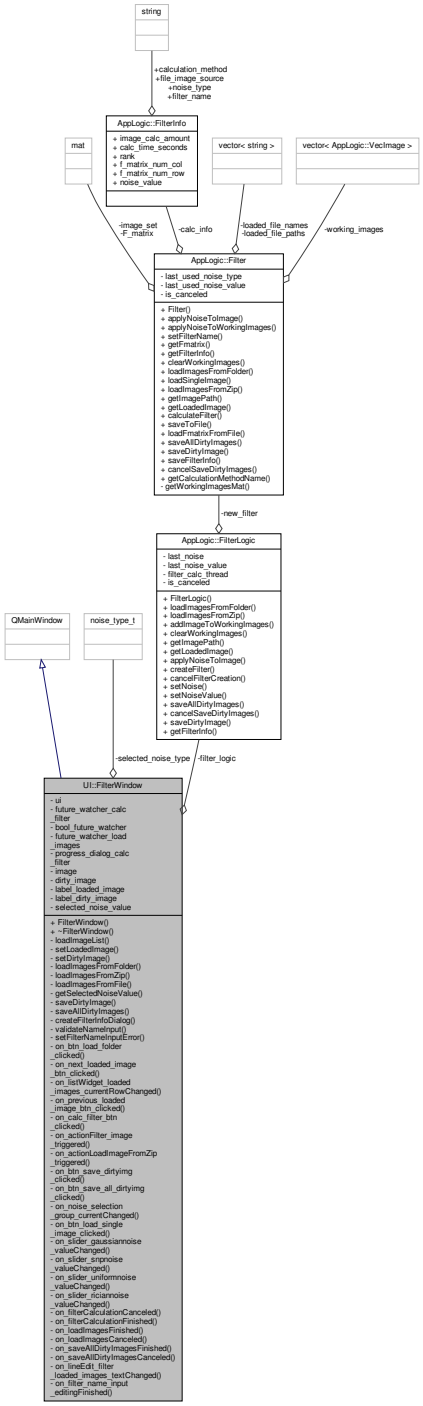
6.4 UI::FilterWindow Class Reference

Window to create filters.

```
#include <FilterWindow.h>
```

Inheritance diagram for UI::FilterWindow:

Collaboration diagram for UI::FilterWindow:



Public Member Functions

- FilterWindow (QWidget *parent=nullptr)

  Construct a new FilterWindow object.

- ~FilterWindow ()

  Destroy the FilterWindow object.

Private Slots

- void on_btn_load_folder_clicked ()
- void on_next_loaded_image_btn_clicked ()
- void on_listWidget_loaded_images_currentRowChanged (int currentRow)
- void on_previous_loaded_image_btn_clicked ()
- void on_calc_filter_btn_clicked ()
- void on_actionFilter_image_triggered ()
- void on_actionLoadImageFromZip_triggered ()
- void on_btn_save_dirtyimg_clicked ()
- void on_btn_save_all_dirtyimg_clicked ()
- void on_noise_selection_group_currentChanged (int index)
- void on_btn_load_single_image_clicked ()
- void on_slider_gaussiannoise_valueChanged (int value)
- void on_slider_snpnoise_valueChanged (int value)
- void on_slider_uniformnoise_valueChanged (int value)
- void on_slider_riciannoise_valueChanged (int value)
- void on_filterCalculationCanceled ()
- void on_filterCalculationFinished ()
- void on_loadImagesFinished ()
- void on_loadImagesCanceled ()
- void on_saveAllDirtyImagesFinished ()
- void on_saveAllDirtyImagesCanceled ()
- void on_lineEdit_filter_loaded_images_textChanged (const QString &arg1)
- void on_filter_name_input_editingFinished ()

Private Member Functions

- bool loadImageList (const vector< string > &loaded_filenames)

  sets the image names in the list to display to user
- bool setLoadedImage (const size_t index)

  shows the preview of the loaded image in the window
- bool setDirtyImage ()

  shows the preview of the currently selected loaded image in the window with noise applied
- void loadImagesFromFolder (const QString &folder_path)

  loads images from specified folder and shows them in the window
- void loadImagesFromZip (const QString &folder_path)

  load images from a zip file
- bool loadImagesFromFile (const QStringList &filenames)

  load list of images provided
- size_t getSelectedNoiseValue (AppLogic::noise_type_t noise_type)

  returns the currently selected noise value for the noise type specified
- bool saveDirtyImage (const QString &fileName)

  saves currently displayed noisy image with the selected noise values
- void saveAllDirtyImages (const QString &fileName)

  applies currently selected noise and value to loaded images and saves them to specified folder
- void createFilterInfoDialog ()

  creates a dialog window with the filter creation information
- bool validateNameInput ()

  validates the input for the name of the filter
- void setFilterNameInputError (const QString &error_message)

  changes the color of the filter name input to indicate an error

Private Attributes

- AppLogic::FilterLogic filter_logic
  API to create filters.
- Ui::FilterWindow * ui
  pointer to access most of the window widgets
- QFutureWatcher< bool > future_watcher_calc_filter
  waits for the filter calculation thread to finish
- QFutureWatcher< bool > bool_future_watcher
  used to wait for image saving to finish
- QFutureWatcher< vector< string > > future_watcher_load_images
  used to wait for image loading to finish
- QProgressDialog * progress_dialog_calc_filter
  window to show progress bar
- QImage image
  stores the loaded image
- QImage dirty_image
  stores image with noise applied
- QLabel * label_loaded_image
  label to show loaded image
- QLabel * label_dirty_image
  label to show noisy image
- AppLogic::noise_type_t selected_noise_type
  curent noise type selected
- size_t selected_noise_value
  current noise value

6.4.1 Detailed Description

Window to create filters.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 FilterWindow()

```
UI::FilterWindow::FilterWindow (
            QWidget * parent = nullptr )
```

Construct a new FilterWindow object.

Parameters

parent	parent of the window

6.4.2.2 ~FilterWindow()

```
UI::FilterWindow::~FilterWindow ( )
```

Destroy the FilterWindow object.

6.4.3 Member Function Documentation

6.4.3.1 createFilterInfoDialog()

```
void UI::FilterWindow::createFilterInfoDialog ( )  [private]
```

creates a dialog window with the filter creation information

6.4.3.2 getSelectedNoiseValue()

```
size_t UI::FilterWindow::getSelectedNoiseValue (
            AppLogic::noise_type_t noise_type )  [private]
```

returns the currently selected noise value for the noise type specified

Parameters

noise_type	the type of noise to get the value from

Returns

size_t value input by user for the specified noise type

6.4.3.3 loadImageList()

```
bool UI::FilterWindow::loadImageList (
            const vector< string > & loaded_filenames )  [private]
```

sets the image names in the list to display to user

Parameters

loaded_filenames	names of the loaded images

Returns

true if images were added correctly
false if there was a problem adding images to the list

6.4.3.4 loadImagesFromFile()

```
bool UI::FilterWindow::loadImagesFromFile (
            const QStringList & filenames )  [private]
```

load list of images provided

Parameters

filenames	list of paths of images to load

Returns

true if images are loaded correctly
false if unable to load images

6.4.3.5 loadImagesFromFolder()

```
void UI::FilterWindow::loadImagesFromFolder (
            const QString & folder_path )  [private]
```

loads images from specified folder and shows them in the window

This function will raise a progress dialog window while it waits for images to finish loading

Parameters

folder_path	path of the folder where the images reside

6.4.3.6 loadImagesFromZip()

```
void UI::FilterWindow::loadImagesFromZip (
            const QString & folder_path )  [private]
```

load images from a zip file

This function will raise a progress dialog window while it waits for images to finish loading

Parameters

folder_path	full path of a valid zip file

6.4.3.7 on_actionFilter_image_triggered

```
void UI::FilterWindow::on_actionFilter_image_triggered ( )  [private], [slot]
```

6.4.3.8 on_actionLoadImageFromZip_triggered

```
void UI::FilterWindow::on_actionLoadImageFromZip_triggered ( )  [private], [slot]
```

6.4.3.9 on_btn_load_folder_clicked

```
void UI::FilterWindow::on_btn_load_folder_clicked ( )  [private], [slot]
```

6.4.3.10 on_btn_load_single_image_clicked

```
void UI::FilterWindow::on_btn_load_single_image_clicked ( )  [private], [slot]
```

6.4.3.11 on_btn_save_all_dirtyimg_clicked

```
void UI::FilterWindow::on_btn_save_all_dirtyimg_clicked ( )  [private], [slot]
```

6.4.3.12 on_btn_save_dirtyimg_clicked

```
void UI::FilterWindow::on_btn_save_dirtyimg_clicked ( )  [private], [slot]
```

6.4.3.13 on_calc_filter_btn_clicked

```
void UI::FilterWindow::on_calc_filter_btn_clicked ( )  [private], [slot]
```

6.4.3.14 on_filter_name_input_editingFinished

```
void UI::FilterWindow::on_filter_name_input_editingFinished ( )  [private], [slot]
```

6.4.3.15 on_filterCalculationCanceled

```
void UI::FilterWindow::on_filterCalculationCanceled ( )  [private], [slot]
```

6.4.3.16 on_filterCalculationFinished

```
void UI::FilterWindow::on_filterCalculationFinished ( )  [private], [slot]
```

6.4.3.17 on_lineEdit_filter_loaded_images_textChanged

```
void UI::FilterWindow::on_lineEdit_filter_loaded_images_textChanged (
            const QString & arg1 )  [private], [slot]
```

6.4.3.18 on_listWidget_loaded_images_currentRowChanged

```
void UI::FilterWindow::on_listWidget_loaded_images_currentRowChanged (
            int currentRow )  [private], [slot]
```

6.4.3.19 on_loadImagesCanceled

```
void UI::FilterWindow::on_loadImagesCanceled ( )  [private], [slot]
```

6.4.3.20 on_loadImagesFinished

```
void UI::FilterWindow::on_loadImagesFinished ( )  [private], [slot]
```

6.4.3.21 on_next_loaded_image_btn_clicked

```
void UI::FilterWindow::on_next_loaded_image_btn_clicked ( )  [private], [slot]
```

6.4.3.22 on_noise_selection_group_currentChanged

```
void UI::FilterWindow::on_noise_selection_group_currentChanged (
            int index )  [private], [slot]
```

6.4.3.23 on_previous_loaded_image_btn_clicked

```
void UI::FilterWindow::on_previous_loaded_image_btn_clicked ( )  [private], [slot]
```

6.4.3.24 on_saveAllDirtyImagesCanceled

```
void UI::FilterWindow::on_saveAllDirtyImagesCanceled ( )  [private], [slot]
```

6.4.3.25 on_saveAllDirtyImagesFinished

```
void UI::FilterWindow::on_saveAllDirtyImagesFinished ( )  [private], [slot]
```

6.4.3.26 on_slider_gaussiannoise_valueChanged

```
void UI::FilterWindow::on_slider_gaussiannoise_valueChanged (
            int value )  [private], [slot]
```

6.4.3.27 on_slider_riciannoise_valueChanged

```
void UI::FilterWindow::on_slider_riciannoise_valueChanged (
            int value )  [private], [slot]
```

6.4.3.28 on_slider_snpnoise_valueChanged

```
void UI::FilterWindow::on_slider_snpnoise_valueChanged (
            int value )  [private], [slot]
```

6.4.3.29 on_slider_uniformnoise_valueChanged

```
void UI::FilterWindow::on_slider_uniformnoise_valueChanged (
            int value )  [private], [slot]
```

6.4.3.30 saveAllDirtyImages()

```
void UI::FilterWindow::saveAllDirtyImages (
            const QString & fileName )  [private]
```

applies currently selected noise and value to loaded images and saves them to specified folder

This function will raise a progress dialog window while it waits for images to finish loading

Parameters

fileName	path of the folder where the noisy images will be saved

6.4.3.31 saveDirtyImage()

```
bool UI::FilterWindow::saveDirtyImage (
            const QString & fileName )  [private]
```

saves currently displayed noisy image with the selected noise values

Parameters

fileName	name of the file to save the noisy image

Returns

true if image is saved successfully
false if image saving failed

6.4.3.32 setDirtyImage()

```
bool UI::FilterWindow::setDirtyImage ( )  [private]
```

shows the preview of the currently selected loaded image in the window with noise applied

Returns

true if able to create image and show it
false if unable to create image, clears the image

6.4.3.33 setFilterNameInputError()

```
void UI::FilterWindow::setFilterNameInputError (
            const QString & error_message )  [private]
```

changes the color of the filter name input to indicate an error

Parameters

error_message	error message shown in status bar that explains problem with the filter name

6.4.3.34 setLoadedImage()

```
bool UI::FilterWindow::setLoadedImage (
            const size_t index )  [private]
```

shows the preview of the loaded image in the window

Parameters

index	index of the loaded image to preview

Returns

true if able to create image and show it
false if unable to create image, clears the image

6.4.3.35 validateNameInput()

```
bool UI::FilterWindow::validateNameInput ( )  [private]
```

validates the input for the name of the filter

Checks whether the name exists or if it contains any illegal characters or device names that would make the filename ilegal.

Returns

true if the name of the filter is valid
false if there is a problem with the filtername

6.4.4 Member Data Documentation

6.4.4.1 bool_future_watcher

```
QFutureWatcher<bool> UI::FilterWindow::bool_future_watcher  [private]
```

used to wait for image saving to finish

6.4.4.2 dirty_image

```
QImage UI::FilterWindow::dirty_image  [private]
```

stores image with noise applied

6.4.4.3 filter_logic

```
AppLogic::FilterLogic UI::FilterWindow::filter_logic  [private]
```

API to create filters.

6.4.4.4 future_watcher_calc_filter

```
QFutureWatcher<bool> UI::FilterWindow::future_watcher_calc_filter  [private]
```

waits for the filter calculation thread to finish

6.4.4.5 future_watcher_load_images

```
QFutureWatcher<vector<string> > UI::FilterWindow::future_watcher_load_images  [private]
```

used to wait for image loading to finish

6.4.4.6 image

```
QImage UI::FilterWindow::image  [private]
```

stores the loaded image

6.4.4.7 label_dirty_image

```
QLabel* UI::FilterWindow::label_dirty_image  [private]
```

label to show noisy image

6.4.4.8 label_loaded_image

```
QLabel* UI::FilterWindow::label_loaded_image  [private]
```

label to show loaded image

6.4.4.9 progress_dialog_calc_filter

```
QProgressDialog* UI::FilterWindow::progress_dialog_calc_filter  [private]
```

window to show progress bar

6.4.4.10 selected_noise_type

```
AppLogic::noise_type_t UI::FilterWindow::selected_noise_type  [private]
```

curent noise type selected

6.4.4.11 selected_noise_value

```
size_t UI::FilterWindow::selected_noise_value  [private]
```

current noise value

6.4.4.12 ui

```
Ui::FilterWindow* UI::FilterWindow::ui  [private]
```

pointer to access most of the window widgets

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

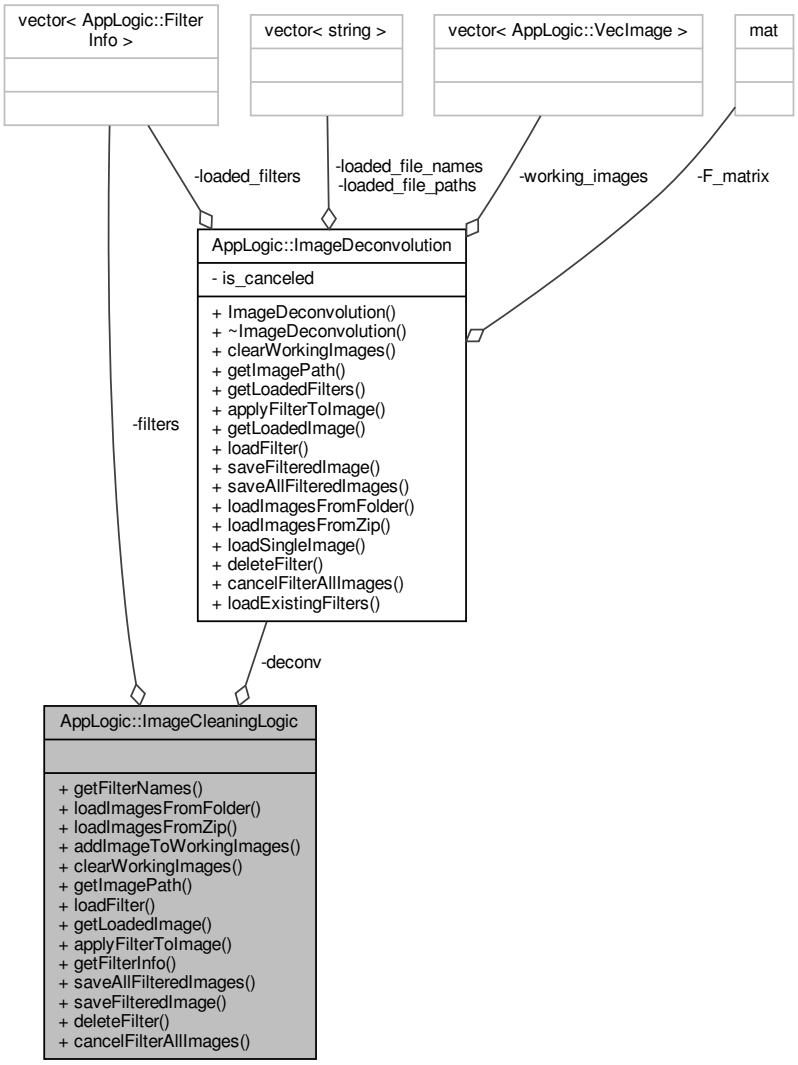
- FilterWindow.h
- FilterWindow.cpp

6.5 AppLogic::ImageCleaningLogic Class Reference

API to the ImageDeconvolution class.

```
#include <ImageCleaningLogic.hpp>
```

Collaboration diagram for AppLogic::ImageCleaningLogic:



Public Member Functions

- vector< string > getFilterNames ()

  returns a list with the names of the filters found on file.
- vector< string > loadImagesFromFolder (string folder_path)

  loads images from specified full path to a folder
- vector< string > loadImagesFromZip (string file_path)

  load images from a zip file
- string addImageToWorkingImages (string file_path)

  loads one image and adds it the the list of working images
- void clearWorkingImages ()

  deletes all loaded images
- string getImagePath (const size_t index)

  returns the path of the loaded image specified by index

- bool loadFilter (int filter_id)

  load filter matrix from file
- VecImage * getLoadedImage (const size_t index)

  returns a pointer to the loaded `VecImage` specified by `index`
- VecImage applyFilterToImage (int image_id)

  returns a new image of the selected image with the filter applied.
- FilterInfo * getFilterInfo (size_t indx)

  returns pointer to the `FilterInfo` object fo current filter.
- bool saveAllFilteredImages (string folder_path)

  applies filter to all working images and saves them in the specified path
- bool saveFilteredImage (size_t image_id, string folder_path)

  applies filter to image and saves it in the specified path
- bool deleteFilter (size_t index)

  deletes filter specified by `index`
- void cancelFilterAllImages ()

  cancels the process of saving images with filter applied

Private Attributes

- ImageDeconvolution deconv

  ImageDeconvolution object that provides functionality.
- vector< FilterInfo > filters

  list of FilterInfo objects containing the information of the filters found on file

6.5.1 Detailed Description

API to the ImageDeconvolution class.

Used to load existing filters and apply filters to loaded images. Allows users to load images, apply already created filters to the images and save those images

6.5.2 Member Function Documentation

6.5.2.1 addImageToWorkingImages()

```
string AppLogic::ImageCleaningLogic::addImageToWorkingImages (
            string file_path )
```

loads one image and adds it the the list of working images

Parameters

file_path	full path of a valid image

Returns

string returns the name of the loaded image

6.5.2.2 applyFilterToImage()

```
VecImage AppLogic::ImageCleaningLogic::applyFilterToImage (
            int image_id )
```

returns a new image of the selected image with the filter applied.

Aplies the current folder to the image identified by `image_id`

Parameters

image_id	index of the image to apply the filter to

Returns

VecImage new `VecImage` with the filter applied

6.5.2.3 cancelFilterAllImages()

```
void AppLogic::ImageCleaningLogic::cancelFilterAllImages ( )
```

cancels the process of saving images with filter applied

6.5.2.4 clearWorkingImages()

```
void AppLogic::ImageCleaningLogic::clearWorkingImages ( )
```

deletes all loaded images

6.5.2.5 deleteFilter()

```
bool AppLogic::ImageCleaningLogic::deleteFilter (
            size_t index )
```

deletes filter specified by `index`

Checks the application data folder in order to delete the ∗.finfo and ∗.mat files that ath the name of the filter specified by `index` in `filters`.

Parameters

index	index of the `FilterInfo` that has the name of the filter to delete

Returns

true if the filter is deleted successfully
false if there is an error deleting the filters

6.5.2.6 getFilterInfo()

```
FilterInfo * AppLogic::ImageCleaningLogic::getFilterInfo (
            size_t indx )
```

returns pointer to the `FilterInfo` object fo current filter.

Returns

FilterInfo∗ Object containing the calculation information for the curent filter

6.5.2.7 getFilterNames()

```
vector< string > AppLogic::ImageCleaningLogic::getFilterNames ( )
```

returns a list with the names of the filters found on file.

Checks the application data folder looking for ∗.finfo files and gets the name of the filters.

Returns

vector<string> list of names of filters found in application data folder

6.5.2.8 getImagePath()

```
string AppLogic::ImageCleaningLogic::getImagePath (
            const size_t index )
```

returns the path of the loaded image specified by index

Parameters

index	the index of the loaded image to get the path for

Returns

string path of the loaded image in the filesystem

6.5.2.9 getLoadedImage()

```
VecImage * AppLogic::ImageCleaningLogic::getLoadedImage (
            const size_t index )
```

returns a pointer to the loaded `VecImage` specified by `index`

Parameters

index	the index of the loaded image to get

Returns

VecImage* pointer to the `VecImage` object of the loaded image

6.5.2.10 loadFilter()

```
bool AppLogic::ImageCleaningLogic::loadFilter (
            int filter_id )
```

load filter matrix from file

Looks for the corresponding *.mat file with the same name as the filter identified by the index `filter_id`, and loads the matrix to be used to apply to images.

Parameters

filter_id	index of the `FilterInfo` in `filters` with the name of the filter to be loaded

Returns

true if the matrix is loaded correctly
false if there is an error loading the matrix

6.5.2.11 loadImagesFromFolder()

```
vector< string > AppLogic::ImageCleaningLogic::loadImagesFromFolder (
            string folder_path )
```

loads images from specified full path to a folder

This function will look for the first valid image in the folder and will load all other valid images of the same resolution as that first loaded image, other images are ignored.

Parameters

folder_path	full path to the directory containing the images

Returns

vector<string> list of names of the images that were loaded

6.5.2.12 loadImagesFromZip()

```
vector< string > AppLogic::ImageCleaningLogic::loadImagesFromZip (
            string file_path )
```

load images from a zip file

This function will go through all files in a zip file. It will look for the first valid image in the folder and will load all other valid images of the same resolution as the first loaded image, other images are ignored.

Parameters

file_path	full path of a valid zip file

Returns

vector<string> list of names of the images that were loaded

6.5.2.13 saveAllFilteredImages()

```
bool AppLogic::ImageCleaningLogic::saveAllFilteredImages (
            string folder_path )
```

applies filter to all working images and saves them in the specified path

Applies filter to all loaded images using the loded filter and saves them to the folder path provided as PNG (.png) images. This operation does not modify the loaded images.

Parameters

folder_path	path where the images will be saved

Returns

true if images are saved correctly
false if an eror is encountered or saving process is canceled

6.5.2.14 saveFilteredImage()

```
bool AppLogic::ImageCleaningLogic::saveFilteredImage (
            size_t image_id,
            string folder_path )
```

applies filter to image and saves it in the specified path

Applies filter to the loaded image specified by the index `image_id`, using the loaded filter and saves it to the folder path provided as a PNG (.png) image. This operation does not modify the loaded image.

Parameters

image_id	index of the image to apply noise to and save
folder_path	name of the file to save the dirty image

Returns

true if image is saved corectly
false if there is an error saving the image

6.5.3 Member Data Documentation

6.5.3.1 deconv

```
ImageDeconvolution AppLogic::ImageCleaningLogic::deconv  [private]
```

ImageDeconvolution object that provides functionality.

6.5.3.2 filters

```
vector<FilterInfo> AppLogic::ImageCleaningLogic::filters  [private]
```

list of FilterInfo objects containing the information of the filters found on file

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

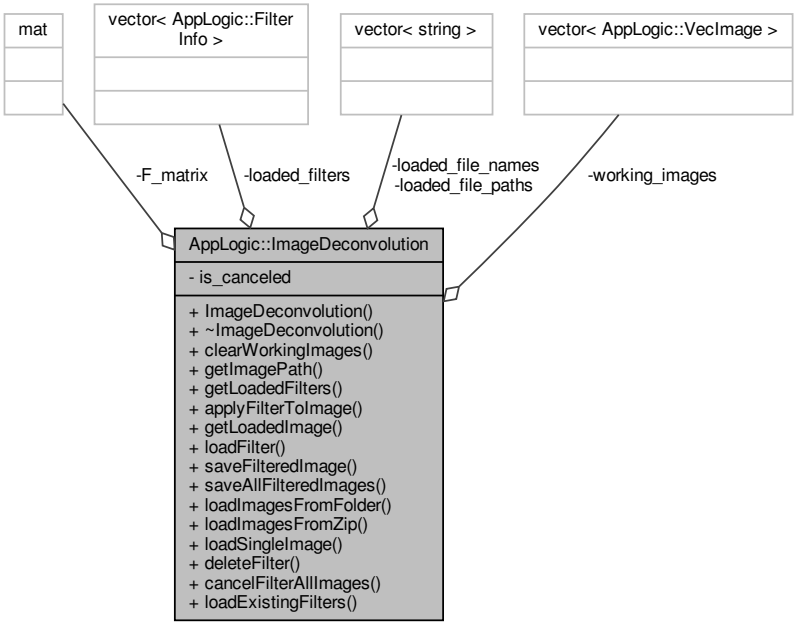
- ImageCleaningLogic.hpp
- ImageCleaningLogic.cpp

6.6 AppLogic::ImageDeconvolution Class Reference

Allows to apply existing filter to images.

`#include <ImageDeconvolution.hpp>`

Collaboration diagram for AppLogic::ImageDeconvolution:



Public Member Functions

- ImageDeconvolution ()

  Construct a new Image Deconvolution object.
- ∼ImageDeconvolution ()

  Destroy the Image Deconvolution object.
- void clearWorkingImages ()

  deletes all loaded image information
- string getImagePath (size_t index)

  returns the path of the loaded image specified by index
- vector< FilterInfo > getLoadedFilters ()

  Get the Loaded Filters object.
- VecImage applyFilterToImage (size_t image_id)

  returns a new image of the selected image with the filter applied.
- VecImage * getLoadedImage (const size_t index)

  returns a pointer to the loaded `VecImage` specified by `index`
- bool loadFilter (size_t id)

  load filter matrix from file
- bool saveFilteredImage (size_t image_id, string folder_path)

  applies filter to image and saves it in the specified path

- bool saveAllFilteredImages (string folder_path)

  applies filter to all working images and saves them in the specified path
- vector< string > loadImagesFromFolder (string folder_path)

  loads images from specified full path to a folder
- vector< string > loadImagesFromZip (string file_path)

  load images from a zip file
- string loadSingleImage (string file_path)

  loads a single image and adds it to the working images
- bool deleteFilter (size_t index)

  deletes filter specified by `index`
- void cancelFilterAllImages ()

  cancels the process of saving images with filter applied
- vector< FilterInfo > loadExistingFilters ()

  checks application data directory in order to load existing filter information

Private Attributes

- vector< FilterInfo > loaded_filters

  filter information of existing filters
- vector< VecImage > working_images

  list of loaded images
- vector< string > loaded_file_paths

  paths to the loaded image files
- vector< string > loaded_file_names

  names of the loaded images
- bool is_canceled

  indicates whether image saving process is canceled
- mat F_matrix

  Filter matrix.

6.6.1 Detailed Description

Allows to apply existing filter to images.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 ImageDeconvolution()

```
AppLogic::ImageDeconvolution::ImageDeconvolution ( )
```

Construct a new Image Deconvolution object.

6.6.2.2 ~ImageDeconvolution()

```
AppLogic::ImageDeconvolution::~ImageDeconvolution ( )
```

Destroy the Image Deconvolution object.

6.6.3 Member Function Documentation

6.6.3.1 applyFilterToImage()

```
VecImage AppLogic::ImageDeconvolution::applyFilterToImage (
            size_t image_id )
```

returns a new image of the selected image with the filter applied.

Aplies the current folder to the image identified by `image_id`

Parameters

image_id	index of the image to apply the filter to

Returns

VecImage new `VecImage` with the filter applied

6.6.3.2 cancelFilterAllImages()

```
void AppLogic::ImageDeconvolution::cancelFilterAllImages ( )
```

cancels the process of saving images with filter applied

6.6.3.3 clearWorkingImages()

```
void AppLogic::ImageDeconvolution::clearWorkingImages ( )  [inline]
```

deletes all loaded image information

6.6.3.4 deleteFilter()

```
bool AppLogic::ImageDeconvolution::deleteFilter (
            size_t index )
```

deletes filter specified by `index`

Checks the application data folder in order to delete the ∗.finfo and ∗.mat files that ath the name of the filter specified by `index` in `filters`.

Parameters

index	index of the `FilterInfo` that has the name of the filter to delete

Returns

true if the filter is deleted successfully
false if there is an error deleting the filters

6.6.3.5 getImagePath()

```
string AppLogic::ImageDeconvolution::getImagePath (
            size_t index )
```

returns the path of the loaded image specified by index

Parameters

index	the index of the loaded image to get the path for

Returns

string path of the loaded image in the filesystem

6.6.3.6 getLoadedFilters()

```
vector<FilterInfo> AppLogic::ImageDeconvolution::getLoadedFilters ( )  [inline]
```

Get the Loaded Filters object.

Returns

vector<FilterInfo> list of FilterInfo objects containing the information of the existing filters

6.6.3.7 getLoadedImage()

```
VecImage * AppLogic::ImageDeconvolution::getLoadedImage (
            const size_t index )
```

returns a pointer to the loaded `VecImage` specified by `index`

Parameters

index	the index of the loaded image to get

Returns

VecImage* pointer to the VecImage object of the loaded image

6.6.3.8 loadExistingFilters()

```
vector< FilterInfo > AppLogic::ImageDeconvolution::loadExistingFilters ( )
```

checks application data directory in order to load existing filter information

Returns

vector<FilterInfo> list of FilterInfo objects for existing filters in filesystem

6.6.3.9 loadFilter()

```
bool AppLogic::ImageDeconvolution::loadFilter (
            size_t id )
```

load filter matrix from file

Looks for the corresponding *.mat file with the same name as the filter identified by the index `id`, and loads the matrix to be used to apply to images.

Parameters

id	index of the `FilterInfo` in `loaded_filters` with the name of the filter to be loaded

Returns

true if the matrix is loaded correctly
false if there is an error loading the matrix

6.6.3.10 loadImagesFromFolder()

```
vector< string > AppLogic::ImageDeconvolution::loadImagesFromFolder (
            string folder_path )
```

loads images from specified full path to a folder

This function will look for the first valid image in the folder and will load all other valid images of the same resolution as that first loaded image, other images are ignored.

Parameters

folder_path	full path to the directory containing the images

Returns

vector<string> list of names of the images that were loaded

6.6.3.11 loadImagesFromZip()

```
vector< string > AppLogic::ImageDeconvolution::loadImagesFromZip (
            string file_path )
```

load images from a zip file

This function will go through all files in a zip file. It will look for the first valid image in the folder and will load all other valid images of the same resolution as the first loaded image, other images are ignored.

Parameters

file_path	full path of a valid zip file

Returns

vector<string> list of names of the images that were loaded

6.6.3.12 loadSingleImage()

```
string AppLogic::ImageDeconvolution::loadSingleImage (
            string file_path )
```

loads a single image and adds it to the working images

Parameters

file_path	path of the image file

Returns

string name of the loaded image

6.6.3.13 saveAllFilteredImages()

```
bool AppLogic::ImageDeconvolution::saveAllFilteredImages (
            string folder_path )
```

applies filter to all working images and saves them in the specified path

Applies filter to all loaded images using the loded filter and saves them to the folder path provided as PNG (.png) images. This operation does not modify the loaded images.

Parameters

folder_path	path where the images will be saved

Returns

true if images are saved correctly
false if an eror is encountered or saving process is canceled

6.6.3.14 saveFilteredImage()

```
bool AppLogic::ImageDeconvolution::saveFilteredImage (
            size_t image_id,
            string folder_path )
```

applies filter to image and saves it in the specified path

Applies filter to the loaded image specified by the index `image_id`, using the loaded filter and saves it to the folder path provided as a PNG (.png) image. This operation does not modify the loaded image.

Parameters

image_id	index of the image to apply noise to and save
folder_path	name of the file to save the dirty image

Returns

true if image is saved corectly
false if there is an error saving the image

6.6.4 Member Data Documentation

6.6.4.1 F_matrix

```
mat AppLogic::ImageDeconvolution::F_matrix  [private]
```

Filter matrix.

6.6.4.2 is_canceled

```
bool AppLogic::ImageDeconvolution::is_canceled  [private]
```

indicates whether image saving process is canceled

6.6.4.3 loaded_file_names

```
vector<string> AppLogic::ImageDeconvolution::loaded_file_names  [private]
```

names of the loaded images

6.6.4.4 loaded_file_paths

```
vector<string> AppLogic::ImageDeconvolution::loaded_file_paths  [private]
```

paths to the loaded image files

6.6.4.5 loaded_filters

```
vector<FilterInfo> AppLogic::ImageDeconvolution::loaded_filters  [private]
```

filter information of existing filters

6.6.4.6 working_images

```
vector<VecImage> AppLogic::ImageDeconvolution::working_images  [private]
```

list of loaded images

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

- ImageDeconvolution.hpp
- ImageDeconvolution.cpp

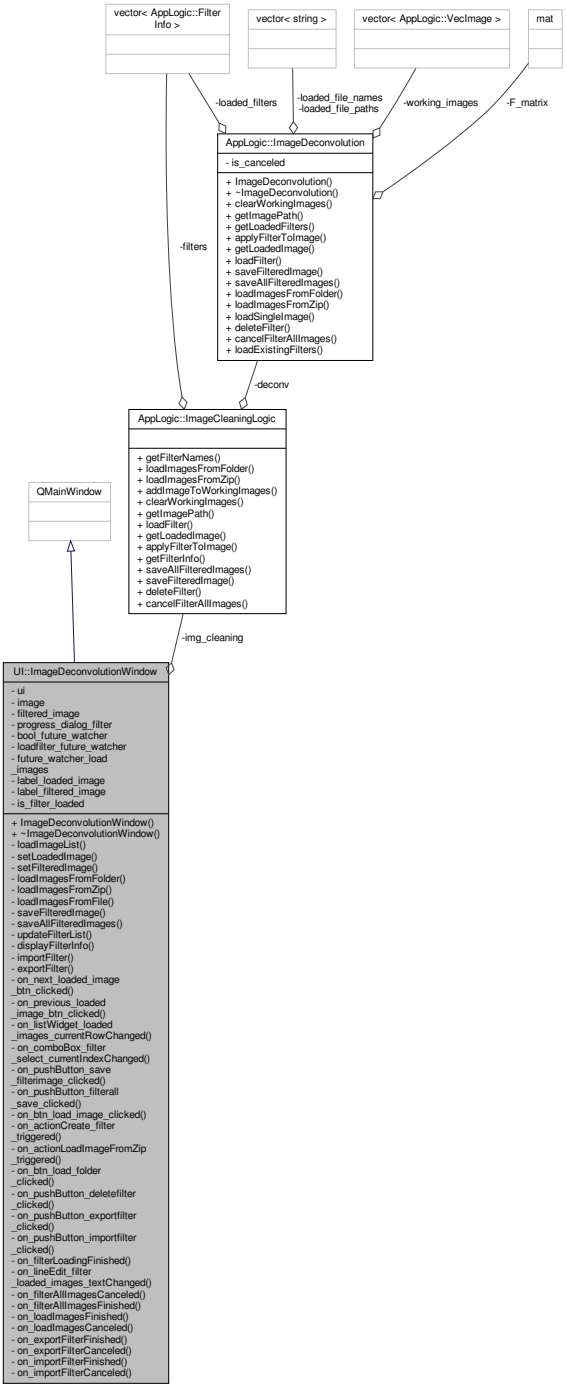
6.7 UI::ImageDeconvolutionWindow Class Reference

Window to filter images.

`#include <ImageDeconvolutionWindow.h>`

Inheritance diagram for UI::ImageDeconvolutionWindow:

Collaboration diagram for UI::ImageDeconvolutionWindow:



Public Member Functions

- ImageDeconvolutionWindow (QWidget *parent=nullptr)

  Construct a new Image Deconvolution Window object.

- ~ImageDeconvolutionWindow ()

  Destroy the Image Deconvolution Window object.

Private Slots

- void on_next_loaded_image_btn_clicked ()

  functions called whenever there is an event on the window
- void on_previous_loaded_image_btn_clicked ()
- void on_listWidget_loaded_images_currentRowChanged (int currentRow)
- void on_comboBox_filter_select_currentIndexChanged (int index)
- void on_pushButton_save_filterimage_clicked ()
- void on_pushButton_filterall_save_clicked ()
- void on_btn_load_image_clicked ()
- void on_actionCreate_filter_triggered ()
- void on_actionLoadImageFromZip_triggered ()
- void on_btn_load_folder_clicked ()
- void on_pushButton_deletefilter_clicked ()
- void on_pushButton_exportfilter_clicked ()
- void on_pushButton_importfilter_clicked ()
- void on_filterLoadingFinished ()
- void on_lineEdit_filter_loaded_images_textChanged (const QString &arg1)
- void on_filterAllImagesCanceled ()
- void on_filterAllImagesFinished ()
- void on_loadImagesFinished ()
- void on_loadImagesCanceled ()
- void on_exportFilterFinished ()
- void on_exportFilterCanceled ()
- void on_importFilterFinished ()
- void on_importFilterCanceled ()

Private Member Functions

- bool loadImageList (const std::vector< string > &loaded_filenames)

  sets the image names in the list to display to user
- bool setLoadedImage (const size_t index)

  shows the preview of the loaded image in the window
- bool setFilteredImage (const size_t index)

  shows the preview of the currently selected loaded image in the window with filter applied
- void loadImagesFromFolder (const QString &folder_path)

  loads images from specified folder and shows them in the window
- void loadImagesFromZip (const QString &folder_path)

  load images from a zip file
- bool loadImagesFromFile (const QStringList &filenames)

  load list of images provided
- bool saveFilteredImage (const QString &fileName)

  saves currently displayed filter image
- void saveAllFilteredImages (const QString &fileName)

  applies currently selected filter to loaded images and saves them to specified folder
- void updateFilterList ()

  updates the list of loaded filters
- void displayFilterInfo (AppLogic::FilterInfo *fi)

  displays the filter information on the window
- void importFilter (const QString &import_filename)

  imports filter from a zip export file
- void exportFilter (const QString &export_filename)

  exports currently selected filter as a zip file

Private Attributes

- Ui::ImageDeconvolutionWindow * ui
  pointer to access most of the window widgets
- AppLogic::ImageCleaningLogic img_cleaning
  API to filter images.
- QImage image
  stores the loaded image
- QImage filtered_image
  stores the image with filter applied
- QProgressDialog * progress_dialog_filter
  window to show progress bar
- QFutureWatcher< bool > bool_future_watcher
  used to wait for image saving to finish
- QFutureWatcher< bool > loadfilter_future_watcher
  used to wait for filters to load
- QFutureWatcher< vector< string > > future_watcher_load_images
  used to wait for image loading to finish
- QLabel * label_loaded_image
  label to show loaded image
- QLabel * label_filtered_image
  label to show filtered image
- bool is_filter_loaded
  label to show loaded image

6.7.1 Detailed Description

Window to filter images.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 ImageDeconvolutionWindow()

```
UI::ImageDeconvolutionWindow::ImageDeconvolutionWindow (
            QWidget * parent = nullptr )  [explicit]
```

Construct a new Image Deconvolution Window object.

Parameters

parent	parent of the window

6.7.2.2 ~ImageDeconvolutionWindow()

```
UI::ImageDeconvolutionWindow::~ImageDeconvolutionWindow ( )
```

Destroy the Image Deconvolution Window object.

6.7.3 Member Function Documentation

6.7.3.1 displayFilterInfo()

```
void UI::ImageDeconvolutionWindow::displayFilterInfo (
            AppLogic::FilterInfo * fi )  [private]
```

displays the filter information on the window

Parameters

fi	`FilterInfo` object containing the information about the filter

6.7.3.2 exportFilter()

```
void UI::ImageDeconvolutionWindow::exportFilter (
            const QString & export_filename )  [private]
```

exports currently selected filter as a zip file

Parameters

export_filename	

6.7.3.3 importFilter()

```
void UI::ImageDeconvolutionWindow::importFilter (
            const QString & import_filename )  [private]
```

imports filter from a zip export file

Parameters

import_filename	path to the zip file export

6.7.3.4 loadImageList()

```
bool UI::ImageDeconvolutionWindow::loadImageList (
            const std::vector< string > & loaded_filenames )  [private]
```

sets the image names in the list to display to user

Parameters

loaded_filenames	names of the loaded images

Returns

true if images were added correctly
false if there was a problem adding images to the list

6.7.3.5 loadImagesFromFile()

```
bool UI::ImageDeconvolutionWindow::loadImagesFromFile (
            const QStringList & filenames )  [private]
```

load list of images provided

Parameters

filenames	list of paths of images to load

Returns

true if images are loaded correctly
false if unable to load images

6.7.3.6 loadImagesFromFolder()

```
void UI::ImageDeconvolutionWindow::loadImagesFromFolder (
            const QString & folder_path )  [private]
```

loads images from specified folder and shows them in the window

This function will raise a progress dialog window while it waits for images to finish loading

Parameters

folder_path	path of the folder where the images reside

6.7.3.7 loadImagesFromZip()

```
void UI::ImageDeconvolutionWindow::loadImagesFromZip (
            const QString & folder_path )  [private]
```

load images from a zip file

This function will raise a progress dialog window while it waits for images to finish loading

Parameters

folder_path	full path of a valid zip file

6.7.3.8 on_actionCreate_filter_triggered

```
void UI::ImageDeconvolutionWindow::on_actionCreate_filter_triggered ( )  [private], [slot]
```

6.7.3.9 on_actionLoadImageFromZip_triggered

```
void UI::ImageDeconvolutionWindow::on_actionLoadImageFromZip_triggered ( )  [private], [slot]
```

6.7.3.10 on_btn_load_folder_clicked

```
void UI::ImageDeconvolutionWindow::on_btn_load_folder_clicked ( )  [private], [slot]
```

6.7.3.11 on_btn_load_image_clicked

```
void UI::ImageDeconvolutionWindow::on_btn_load_image_clicked ( )  [private], [slot]
```

6.7.3.12 on_comboBox_filter_select_currentIndexChanged

```
void UI::ImageDeconvolutionWindow::on_comboBox_filter_select_currentIndexChanged (
            int index )  [private], [slot]
```

6.7.3.13 on_exportFilterCanceled

```
void UI::ImageDeconvolutionWindow::on_exportFilterCanceled ( )  [private], [slot]
```

6.7.3.14 on_exportFilterFinished

```
void UI::ImageDeconvolutionWindow::on_exportFilterFinished ( )  [private], [slot]
```

6.7.3.15 on_filterAllImagesCanceled

```
void UI::ImageDeconvolutionWindow::on_filterAllImagesCanceled ( )  [private], [slot]
```

6.7.3.16 on_filterAllImagesFinished

```
void UI::ImageDeconvolutionWindow::on_filterAllImagesFinished ( )  [private], [slot]
```

6.7.3.17 on_filterLoadingFinished

```
void UI::ImageDeconvolutionWindow::on_filterLoadingFinished ( )  [private], [slot]
```

6.7.3.18 on_importFilterCanceled

```
void UI::ImageDeconvolutionWindow::on_importFilterCanceled ( )  [private], [slot]
```

6.7.3.19 on_importFilterFinished

```
void UI::ImageDeconvolutionWindow::on_importFilterFinished ( )  [private], [slot]
```

6.7.3.20 on_lineEdit_filter_loaded_images_textChanged

```
void UI::ImageDeconvolutionWindow::on_lineEdit_filter_loaded_images_textChanged (
            const QString & arg1 )  [private], [slot]
```

6.7.3.21 on_listWidget_loaded_images_currentRowChanged

```
void UI::ImageDeconvolutionWindow::on_listWidget_loaded_images_currentRowChanged (
            int currentRow )  [private], [slot]
```

6.7.3.22 on_loadImagesCanceled

```
void UI::ImageDeconvolutionWindow::on_loadImagesCanceled ( )  [private], [slot]
```

6.7.3.23 on_loadImagesFinished

```
void UI::ImageDeconvolutionWindow::on_loadImagesFinished ( )  [private], [slot]
```

6.7.3.24 on_next_loaded_image_btn_clicked

```
void UI::ImageDeconvolutionWindow::on_next_loaded_image_btn_clicked ( )  [private], [slot]
```

functions called whenever there is an event on the window

These methods are self explanatory from the name.

6.7.3.25 on_previous_loaded_image_btn_clicked

```
void UI::ImageDeconvolutionWindow::on_previous_loaded_image_btn_clicked ( )  [private], [slot]
```

6.7.3.26 on_pushButton_deletefilter_clicked

```
void UI::ImageDeconvolutionWindow::on_pushButton_deletefilter_clicked ( )  [private], [slot]
```

6.7.3.27 on_pushButton_exportfilter_clicked

```
void UI::ImageDeconvolutionWindow::on_pushButton_exportfilter_clicked ( )  [private], [slot]
```

6.7.3.28 on_pushButton_filterall_save_clicked

```
void UI::ImageDeconvolutionWindow::on_pushButton_filterall_save_clicked ( )  [private], [slot]
```

6.7.3.29 on_pushButton_importfilter_clicked

```
void UI::ImageDeconvolutionWindow::on_pushButton_importfilter_clicked ( )  [private], [slot]
```

6.7.3.30 on_pushButton_save_filterimage_clicked

```
void UI::ImageDeconvolutionWindow::on_pushButton_save_filterimage_clicked ( )  [private],
[slot]
```

6.7.3.31 saveAllFilteredImages()

```
void UI::ImageDeconvolutionWindow::saveAllFilteredImages (
            const QString & fileName )  [private]
```

applies currently selected filter to loaded images and saves them to specified folder

This function will raise a progress dialog window while it waits for images to finish saving

Parameters

fileName	path of the folder where the filtered images will be saved

6.7.3.32 saveFilteredImage()

```
bool UI::ImageDeconvolutionWindow::saveFilteredImage (
            const QString & fileName )  [private]
```

saves currently displayed filter image

Parameters

fileName	name of the file to save the filtered image

Returns

true if image is saved successfully
false if image saving failed

6.7.3.33 setFilteredImage()

```
bool UI::ImageDeconvolutionWindow::setFilteredImage (
            const size_t index )  [private]
```

shows the preview of the currently selected loaded image in the window with filter applied

Parameters

index	index of the image to filter

Returns

true if able to create image and show it
false if unable to create image, clears the image

6.7.3.34 setLoadedImage()

```
bool UI::ImageDeconvolutionWindow::setLoadedImage (
            const size_t index )  [private]
```

shows the preview of the loaded image in the window

Parameters

index	index of the loaded image to preview

Returns

true if able to create image and show it
false if unable to create image, clears the image

6.7.3.35 updateFilterList()

```
void UI::ImageDeconvolutionWindow::updateFilterList ( )  [private]
```

updates the list of loaded filters

6.7.4 Member Data Documentation

6.7.4.1 bool_future_watcher

```
QFutureWatcher<bool> UI::ImageDeconvolutionWindow::bool_future_watcher  [private]
```

used to wait for image saving to finish

6.7.4.2 filtered_image

```
QImage UI::ImageDeconvolutionWindow::filtered_image  [private]
```

stores the image with filter applied

6.7.4.3 future_watcher_load_images

```
QFutureWatcher<vector<string> > UI::ImageDeconvolutionWindow::future_watcher_load_images
[private]
```

used to wait for image loading to finish

6.7.4.4 image

```
QImage UI::ImageDeconvolutionWindow::image  [private]
```

stores the loaded image

6.7.4.5 img_cleaning

```
AppLogic::ImageCleaningLogic UI::ImageDeconvolutionWindow::img_cleaning [private]
```

API to filter images.

6.7.4.6 is_filter_loaded

```
bool UI::ImageDeconvolutionWindow::is_filter_loaded [private]
```

label to show loaded image

6.7.4.7 label_filtered_image

```
QLabel* UI::ImageDeconvolutionWindow::label_filtered_image [private]
```

label to show filtered image

6.7.4.8 label_loaded_image

```
QLabel* UI::ImageDeconvolutionWindow::label_loaded_image [private]
```

label to show loaded image

6.7.4.9 loadfilter_future_watcher

```
QFutureWatcher<bool> UI::ImageDeconvolutionWindow::loadfilter_future_watcher [private]
```

used to wait for filters to load

6.7.4.10 progress_dialog_filter

```
QProgressDialog* UI::ImageDeconvolutionWindow::progress_dialog_filter [private]
```

window to show progress bar

6.7.4.11 ui

```
Ui::ImageDeconvolutionWindow* UI::ImageDeconvolutionWindow::ui  [private]
```

pointer to access most of the window widgets

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

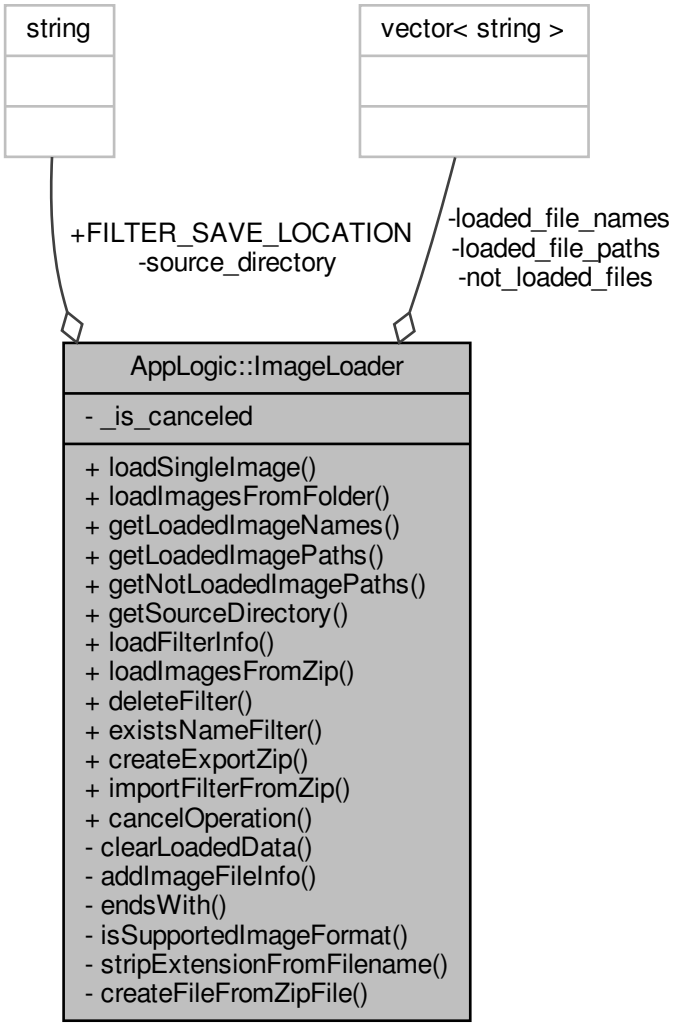
- ImageDeconvolutionWindow.h
- ImageDeconvolutionWindow.cpp

6.8 AppLogic::ImageLoader Class Reference

Allows to load images and filter information as well as import and export filters.

```
#include <ImageLoader.hpp>
```

Collaboration diagram for AppLogic::ImageLoader:



Public Member Functions

- VecImage loadSingleImage (const char *filename)

  loads image from file and returns a VecImage
- vector< VecImage > loadImagesFromFolder (const char *folder_path)

  loads images from specified full path to a folder
- vector< string > getLoadedImageNames ()

  returns the list of the loaded image names
- vector< string > getLoadedImagePaths ()

  returns the list of the loaded image paths
- vector< string > getNotLoadedImagePaths ()

  returns the list of image names that were not loaded
- string getSourceDirectory ()

  returns the source directory from where the images where loaded
- vector< FilterInfo > loadFilterInfo ()

  checks the app's data folder for existing filter info files
- vector< VecImage > loadImagesFromZip (string file_path)

  load images from a zip file

Static Public Member Functions

- static bool deleteFilter (const string filter_name)

  deletes filter from the app's data folder
- static bool existsNameFilter (string filter_name)

  checks if a filter with the provided name already exists in the app's data folder
- static bool createExportZip (string export_filename, string filter_name)

  creates an export Zip file
- static bool importFilterFromZip (string import_filename)

  imports an exported filter from a zip file
- static void cancelOperation (bool status)

  sets cancel status

Static Public Attributes

- static const string FILTER_SAVE_LOCATION = string(getenv("HOME")) + "/.fsp_imgdcnv/"

  Application's data folder in Linux filesystem.

Private Member Functions

- void clearLoadedData ()

  clears all loaded information
- void addImageFileInfo (const string &full_path)

  adds image information to the loaded images lists

Static Private Member Functions

- static bool endsWith (const std::string &str, const std::string &suffix)

  checks whether a string ends with supplied suffinx
- static bool isSupportedImageFormat (const string &str)

  checks whether a provided image filename is a supported format
- static string stripExtensionFromFilename (const string &filename)

  removes file extension from filename
- static bool createFileFromZipFile (string filename, fstream &stream, zip_file *zip_file)

  Create a file from Zip file object.

Private Attributes

- vector< string > loaded_file_paths

  path of the files for the loaded images
- vector< string > loaded_file_names

  names of the loaded images
- vector< string > not_loaded_files

  list of not loaded image names
- string source_directory

  name of the folder from where the images where loaded

Static Private Attributes

- static bool _is_canceled = false

  indicates whether image loading/import/export operation has been canceled

6.8.1 Detailed Description

Allows to load images and filter information as well as import and export filters.

6.8.2 Member Function Documentation

6.8.2.1 addImageFileInfo()

```
void AppLogic::ImageLoader::addImageFileInfo (
            const string & full_path )  [private]
```

adds image information to the loaded images lists

Parameters

full_path	path of the image file

6.8.2.2 cancelOperation()

```
void AppLogic::ImageLoader::cancelOperation (
            bool status )  [static]
```

sets cancel status

Cancels the loading operations and the export/import operations.

Parameters

status	cancel status

6.8.2.3 clearLoadedData()

```
void AppLogic::ImageLoader::clearLoadedData ( )  [inline], [private]
```

clears all loaded information

6.8.2.4 createExportZip()

```
bool AppLogic::ImageLoader::createExportZip (
            string export_filename,
            string filter_name )  [static]
```

creates an export Zip file

Creates a zip file with both the ∗.mat and ∗.finfo files that contain the filter information

Parameters

export_filename	name of the zip file to create
filter_name	name of the filter to be exported

Returns

true if filter is exported successfully
false if there is a problem exporting the filter

6.8.2.5 createFileFromZipFile()

```
bool AppLogic::ImageLoader::createFileFromZipFile (
            string filename,
            fstream & stream,
            zip_file * zip_file )  [static], [private]
```

Create a file from Zip file object.

Parameters

filename	
stream	
zip_file	

Returns

true
false

6.8.2.6 deleteFilter()

```
bool AppLogic::ImageLoader::deleteFilter (
            const string filter_name )  [static]
```

deletes filter from the app's data folder

This deletion is linux specific since it uses OS directives to remove the files

Parameters

filter_name	name of the filter to delete

Returns

true if the deletion is successful
false if there is a problem deleting the filter

6.8.2.7 endsWith()

```
static bool AppLogic::ImageLoader::endsWith (
            const std::string & str,
            const std::string & suffix )  [inline], [static], [private]
```

checks whether a string ends with supplied suffinx

Parameters

str	string to check
suffix	suffix to check

Returns

true if string ends in suffix
false is string ends in something different than suffix

6.8.2.8 existsNameFilter()

```
bool AppLogic::ImageLoader::existsNameFilter (
            string filter_name )  [static]
```

checks if a filter with the provided name already exists in the app's data folder

Parameters

filter_name	name of the filter to check

Returns

true if a filter with the same name already exists
false if there is no filter with the same name

6.8.2.9 getLoadedImageNames()

```
vector< string > AppLogic::ImageLoader::getLoadedImageNames ( )
```

returns the list of the loaded image names

Returns

vector<string> list of the loaded image names

6.8.2.10 getLoadedImagePaths()

```
vector< string > AppLogic::ImageLoader::getLoadedImagePaths ( )
```

returns the list of the loaded image paths

Returns

vector<string> list of the loaded image paths

6.8.2.11 getNotLoadedImagePaths()

```
vector< string > AppLogic::ImageLoader::getNotLoadedImagePaths ( )
```

returns the list of image names that were not loaded

Returns

vector<string>

6.8.2.12 getSourceDirectory()

```
string AppLogic::ImageLoader::getSourceDirectory ( )
```

returns the source directory from where the images where loaded

Returns

string source directory from where the images where loaded

6.8.2.13 importFilterFromZip()

```
bool AppLogic::ImageLoader::importFilterFromZip (
            string import_filename )  [static]
```

imports an exported filter from a zip file

Parameters

import_filename	name of the zip file with the exported filter

Returns

true if filter is imported successfully
false if there is a problem importing the filter

6.8.2.14 isSupportedImageFormat()

```
static bool AppLogic::ImageLoader::isSupportedImageFormat (
            const string & str )  [inline], [static], [private]
```

checks whether a provided image filename is a supported format

Parameters

str	name of the image file

Returns

true if the name of the file ends in one of the supported image format extensions
false if the name of the file does not ends in a supported extension

6.8.2.15 loadFilterInfo()

```
vector< FilterInfo > AppLogic::ImageLoader::loadFilterInfo ( )
```

checks the app's data folder for existing filter info files

Returns

vector<FilterInfo> list of FilterInfo objects found in filesystem

6.8.2.16 loadImagesFromFolder()

```
vector< VecImage > AppLogic::ImageLoader::loadImagesFromFolder (
            const char * folder_path )
```

loads images from specified full path to a folder

This function will look for the first valid image in the folder and will load all other valid images of the same resolution as that first loaded image, other images are ignored.

Parameters

folder_path	full path to the directory containing the images

Returns

vector<VecImage> list of loaded images

6.8.2.17 loadImagesFromZip()

```
vector< VecImage > AppLogic::ImageLoader::loadImagesFromZip (
            string file_path )
```

load images from a zip file

This function will go through all files in a zip file. It will look for the first valid image in the folder and will load all other valid images of the same resolution as the first loaded image, other images are ignored.

Parameters

file_path	full path of a valid zip file

Returns

vector<VecImage> list of loaded images

6.8.2.18 loadSingleImage()

```
VecImage AppLogic::ImageLoader::loadSingleImage (
            const char * filename )
```

loads image from file and returns a VecImage

Parameters

filename	path of the image file

Returns

VecImage loaded image

6.8.2.19 stripExtensionFromFilename()

```
static string AppLogic::ImageLoader::stripExtensionFromFilename (
            const string & filename )  [inline], [static], [private]
```

removes file extension from filename

This function removes everything after the last dot (.) encountered in the filename

Parameters

filename	name of the file with extension

Returns

string name of the file without the extension

6.8.3 Member Data Documentation

6.8.3.1 _is_canceled

```
bool AppLogic::ImageLoader::_is_canceled = false  [static], [private]
```

indicates whether image loading/import/export operation has been canceled

6.8.3.2 FILTER_SAVE_LOCATION

```
const string AppLogic::ImageLoader::FILTER_SAVE_LOCATION = string(getenv("HOME")) + "/.fsp_↩
imgdcnv/"  [static]
```

Application's data folder in Linux filesystem.

6.8.3.3 loaded_file_names

```
vector<string> AppLogic::ImageLoader::loaded_file_names  [private]
```

names of the loaded images

6.8.3.4 loaded_file_paths

```
vector<string> AppLogic::ImageLoader::loaded_file_paths  [private]
```

path of the files for the loaded images

6.8.3.5 not_loaded_files

```
vector<string> AppLogic::ImageLoader::not_loaded_files  [private]
```

list of not loaded image names

6.8.3.6 source_directory

```
string AppLogic::ImageLoader::source_directory  [private]
```

name of the folder from where the images where loaded

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

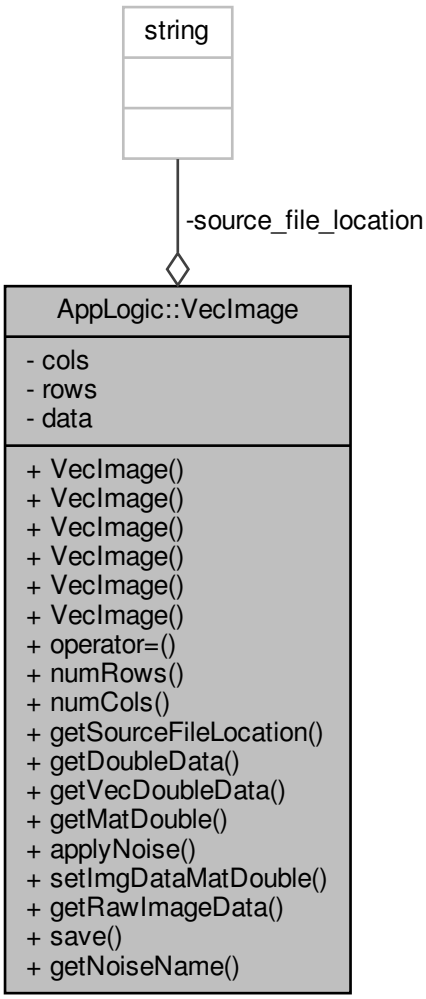
- ImageLoader.hpp
- ImageLoader.cpp

6.9 AppLogic::VecImage Class Reference

Class used for image manipulation.

`#include <VecImage.hpp>`

Collaboration diagram for AppLogic::VecImage:



Public Member Functions

- VecImage ()

  empty image constructor
- VecImage (string filename)

  constructs a new `VecImage` by loading it from file
- VecImage (vector< double > &img_data, size_t width, size_t height)

  construct a new image from an std::vector containing the image data

- VecImage (const vec &img_data, size_t width, size_t height)
  construct a new image from an armadillo::vector containing the image data
- VecImage (CImg< unsigned char > &cImg)
  constructs `VecImage` from a `CImg` object
- VecImage (const VecImage &vImg)
  copy constructor for `VecImage`
- VecImage & operator= (const VecImage &image)
  assignment operator, copies values from one image to the other
- size_t numRows ()
  returns the number of rows (height) of the image
- size_t numCols ()
  returns the number of columns (width) of the image
- string getSourceFileLocation ()
  returns the path from where the image was loaded
- vector< double > getDoubleData ()
  return std::vector with normalized image data
- vec getVecDoubleData ()
  return arma::vector with normalized image data
- mat getMatDouble ()
  returns an arma::matrix with normalized image data
- void applyNoise (double noise_value, noise_type_t noise=noise_type_t::GAUSSIAN)
  applies noise to an image
- void setImgDataMatDouble (const mat &new_data)
  set the data values of the image from an arma::matrix
- const unsigned char * getRawImageData ()
  returns the raw image data
- bool save (const string filename)
  saves an image to file as PNG

Static Public Member Functions

- static string getNoiseName (noise_type_t noise_type)
  returns a `string` name for each of the different noise types

Private Attributes

- string source_file_location
  path of the image in the filesystem when it's loaded from file
- size_t cols
  number of columns in the image
- size_t rows
  number of rows in the image
- cimg_library::CImg< unsigned char > data
  the image data

6.9.1 Detailed Description

Class used for image manipulation.

Allows users to create images, load images from folder, and get the underlying double values of the image in order to calculate filters and apply filters to the images.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 VecImage() [1/6]

```
AppLogic::VecImage::VecImage ( )
```

empty image constructor

6.9.2.2 VecImage() [2/6]

```
AppLogic::VecImage::VecImage (
            string filename )
```

constructs a new VecImage by loading it from file

Loads an image from file and creates a new image in grayscale. Images must be in JPG or PNG format.

Parameters

filename	path of the image to load

6.9.2.3 VecImage() [3/6]

```
AppLogic::VecImage::VecImage (
            vector< double > & img_data,
            size_t width,
            size_t height )
```

construct a new image from an std::vector containing the image data

Constructs a new grayscale image from the `img_data` vector and `width` and `height` specified.

Parameters

img_data	vector containig normalized values [0,1] of the pixel data of the image
width	horizontal size of the image
height	vertical size of the image

6.9.2.4 VecImage() [4/6]

```
AppLogic::VecImage::VecImage (
            const vec & img_data,
            size_t width,
            size_t height )
```

construct a new image from an armadillo::vector containing the image data

Constructs a new grayscale image from the `img_data` vector and `width` and `height` specified.

Parameters

img_data	vector containig normalized values [0,1] of the pixel data of the image
width	horizontal size of the image
height	vertical size of the image

6.9.2.5 VecImage() [5/6]

```
AppLogic::VecImage::VecImage (
            CImg< unsigned char > & cImg )
```

constructs `VecImage` from a `CImg` object

Parameters

cImg	CImg to be used as source for new the image

6.9.2.6 VecImage() [6/6]

```
AppLogic::VecImage::VecImage (
            const VecImage & vImg )
```

copy constructor for `VecImage`

Parameters

vImg	image to copy data from

6.9.3 Member Function Documentation

6.9.3.1 applyNoise()

```
void AppLogic::VecImage::applyNoise (
            double noise_value,
            noise_type_t noise = noise_type_t::GAUSSIAN )
```

applies noise to an image

Parameters

noise_value	value representing the percentage of the noise to be applied [1,99]
noise	type of noise to be applied

6.9.3.2 getDoubleData()

```
std::vector< double > AppLogic::VecImage::getDoubleData ( )
```

return std::vector with normalized image data

Returns

vector<double> std::vector with normalized image data

6.9.3.3 getMatDouble()

```
mat AppLogic::VecImage::getMatDouble ( )
```

returns an arma::matrix with normalized image data

Returns

mat arma::matrix with normalized image data with same size as image

6.9.3.4 getNoiseName()

```
string AppLogic::VecImage::getNoiseName (
            noise_type_t noise_type )  [static]
```

returns a `string` name for each of the different noise types

Parameters

noise_type	type of noise to get the name

Returns

string name of the noise

6.9.3.5 getRawImageData()

```
const unsigned char * AppLogic::VecImage::getRawImageData ( )
```

returns the raw image data

Returns

const unsigned char* image data

6.9.3.6 getSourceFileLocation()

```
string AppLogic::VecImage::getSourceFileLocation ( )
```

returns the path from where the image was loaded

If the image was created in the application and not loaded from file this value will be "internal"

Returns

string path from where the image was loaded

6.9.3.7 getVecDoubleData()

```
vec AppLogic::VecImage::getVecDoubleData ( )
```

return arma::vector with normalized image data

Returns

vec arma::vector with normalized image data

6.9.3.8 numCols()

```
size_t AppLogic::VecImage::numCols ( )
```

returns the number of columns (width) of the image

Returns

size_t number of columns (width) of the image

6.9.3.9 numRows()

```
size_t AppLogic::VecImage::numRows ( )
```

returns the number of rows (height) of the image

Returns

size_t number of rows (height) of the image

6.9.3.10 operator=()

```
VecImage & AppLogic::VecImage::operator= (
            const VecImage & image )
```

assignment operator, copies values from one image to the other

Parameters

image	

Returns

VecImage&

6.9.3.11 save()

```
bool AppLogic::VecImage::save (
            const string filename )
```

saves an image to file as PNG

Parameters

filename	full path of the file where the images will be saved

Returns

true if image is saved succesfully
false if there is a problem saving the image

6.9.3.12 setImgDataMatDouble()

```
void AppLogic::VecImage::setImgDataMatDouble (
            const mat & new_data )
```

set the data values of the image from an arma::matrix

Parameters

new_data	arma::matrix containing image data

6.9.4 Member Data Documentation

6.9.4.1 cols

```
size_t AppLogic::VecImage::cols  [private]
```

number of columns in the image

6.9.4.2 data

```
cimg_library::CImg<unsigned char> AppLogic::VecImage::data  [private]
```

the image data

6.9.4.3 rows

```
size_t AppLogic::VecImage::rows  [private]
```

number of rows in the image

6.9.4.4 source_file_location

```
string AppLogic::VecImage::source_file_location  [private]
```

path of the image in the filesystem when it's loaded from file

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

- VecImage.hpp
- VecImage.cpp

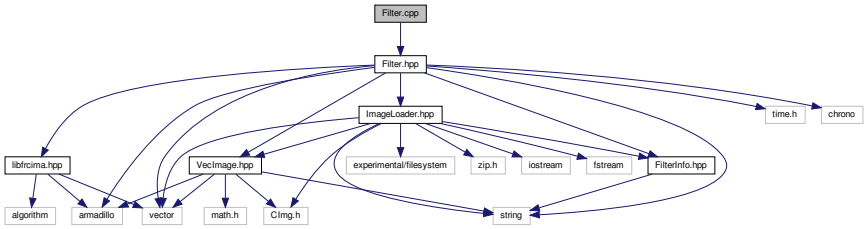
Chapter 7

File Documentation

7.1 Filter.cpp File Reference

```
#include "Filter.hpp"
```

Include dependency graph for Filter.cpp:



Namespaces

- AppLogic

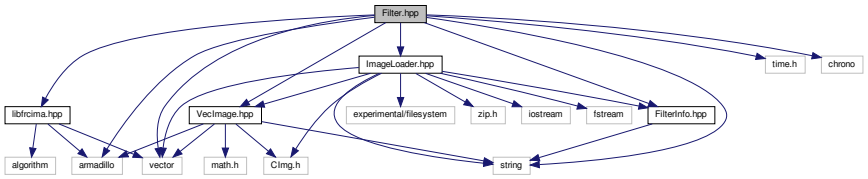
7.2 Filter.hpp File Reference

Contains `Filter` class in charge of creating filters.

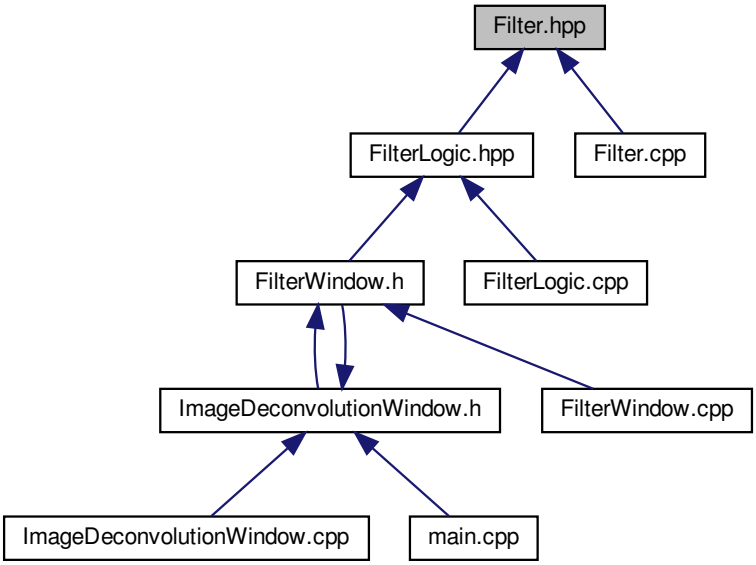
```
#include <vector>
#include <string>
#include <armadillo>
#include <time.h>
#include <chrono>
#include "FilterInfo.hpp"
#include "VecImage.hpp"
#include "ImageLoader.hpp"
```

```
#include "libfrcima.hpp"
```

Include dependency graph for Filter.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AppLogic::Filter

  Allows to create filters based on training sets of images.

Namespaces

- AppLogic

Macros

- #define DIRTY_IMAGE_SUFFIX "_dirty.png"

  suffix used in saved noisy image filenames

Enumerations

- enum AppLogic::calc_method_t { AppLogic::RCIMA_METHOD, AppLogic::FAST_RCIMA_METHOD }
  enum with the calculation methods for the filter

7.2.1 Detailed Description

Contains `Filter` class in charge of creating filters.

Author

Jorge Agüero Zamora

Version

0.1

Date

2021-06-13

7.2.2 Macro Definition Documentation

7.2.2.1 DIRTY_IMAGE_SUFFIX

```
#define DIRTY_IMAGE_SUFFIX "_dirty.png"
```

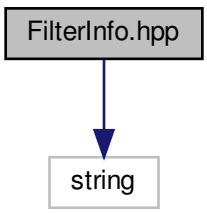
suffix used in saved noisy image filenames

7.3 FilterInfo.hpp File Reference

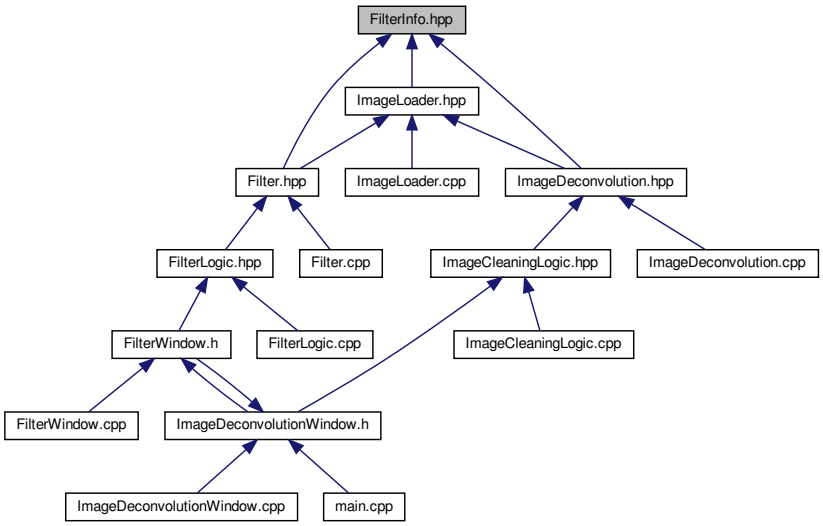
Contains struct to store filter calculation information and the members to save this information into files.

```
#include <string>
```

Include dependency graph for FilterInfo.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct AppLogic::FilterInfo

  struct to hold the the information of the calculated filter

Namespaces

- AppLogic

Functions

- std::ostream & AppLogic::operator<< (std::ostream &stream, FilterInfo const &data)

  outputs filter information to a stream. Used to save FilterInfo to a file.
- std::istream & AppLogic::operator>> (std::istream &stream, FilterInfo &data)

  operator to read filter information from a stream. used to read FilterInfo from file

7.3.1 Detailed Description

Contains struct to store filter calculation information and the members to save this information into files.

Author

Jorge Agüero Zamora

Version

0.1

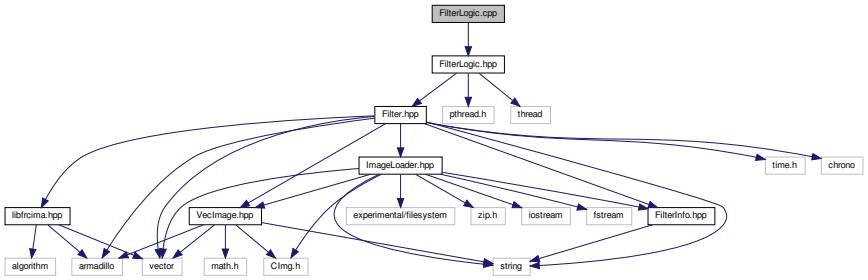
Date

2021-06-13

7.4 FilterLogic.cpp File Reference

```
#include "FilterLogic.hpp"
```

Include dependency graph for FilterLogic.cpp:



Namespaces

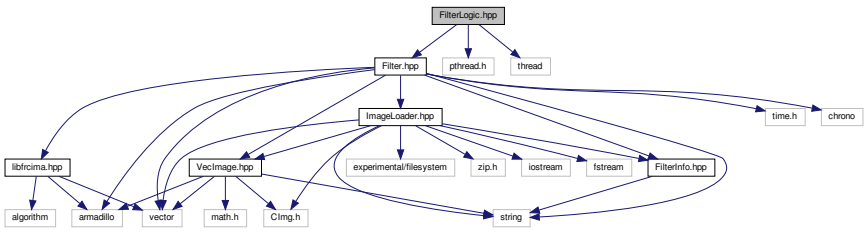
- AppLogic

7.5 FilterLogic.hpp File Reference

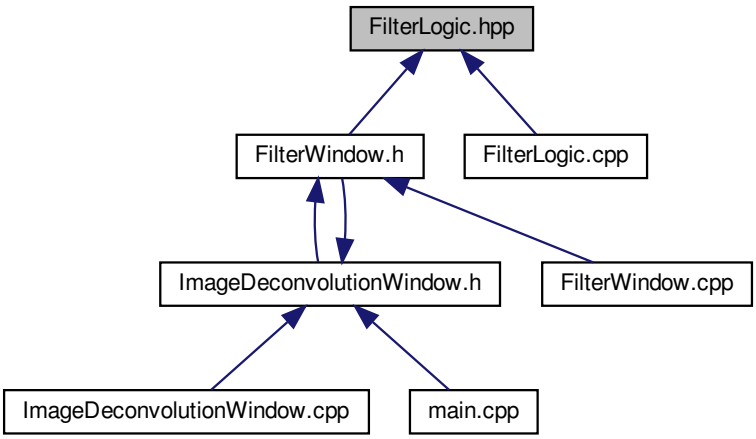
Contains the API for filter creation.

```
#include "Filter.hpp"
#include <pthread.h>
#include <thread>
```

Include dependency graph for FilterLogic.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AppLogic::FilterLogic

  API to the Filter class.

Namespaces

- AppLogic

7.5.1 Detailed Description

Contains the API for filter creation.

Author

Jorge Agüero Zamora

Version

0.1

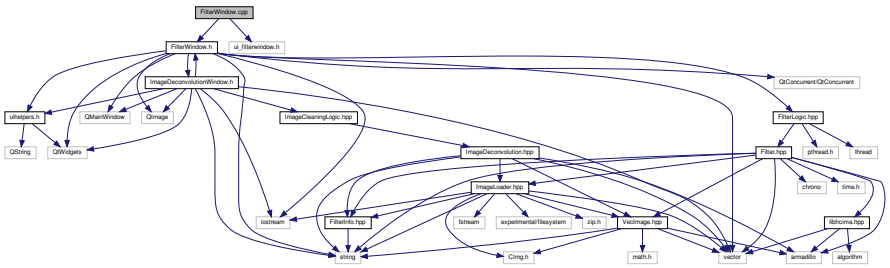
Date

2021-06-13

7.6 FilterWindow.cpp File Reference

```
#include "FilterWindow.h"
#include "ui_filterwindow.h"
```

Include dependency graph for FilterWindow.cpp:



Namespaces

- UI

Macros

- #define WINDOWS_DEVICES "CON|AUX|PRN|COM1|COM2|LPT1|LPT2|NUL"
- #define SLASHES "/\\"

Variables

- static const char UI::notAllowedChars [ ] = "/\\,^@={}[ ]~!?:&*\"|#%<>$\'"();‘’ "
- static const char * UI::notAllowedSubStrings [ ] = {".."}

7.6.1 Macro Definition Documentation

7.6.1.1 SLASHES

```
#define SLASHES "/\\"
```

7.6.1.2 WINDOWS_DEVICES

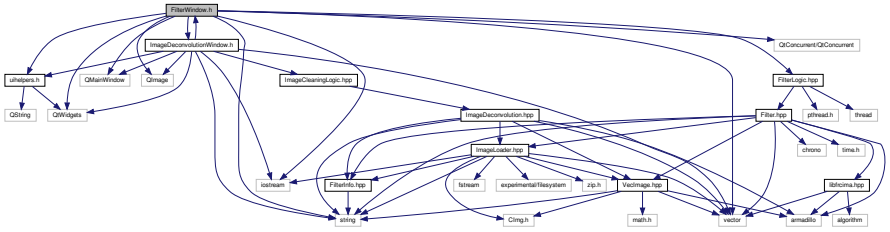
```
#define WINDOWS_DEVICES "CON|AUX|PRN|COM1|COM2|LPT1|LPT2|NUL"
```

7.7 FilterWindow.h File Reference

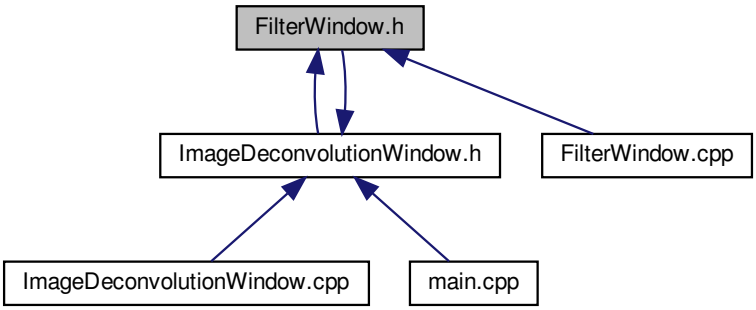
Contains the FilterWindow class which holds the UI to create filters.

```
#include <QMainWindow>
#include <QImage>
#include <QtWidgets>
#include <QtConcurrent/QtConcurrent>
#include <vector>
#include <string>
#include <iostream>
#include "uihelpers.h"
#include "ImageDeconvolutionWindow.h"
#include "FilterLogic.hpp"
```

Include dependency graph for FilterWindow.h:



This graph shows which files directly or indirectly include this file:



Classes

- class UI::FilterWindow

  Window to create filters.

Namespaces

- Ui
- UI

7.7.1 Detailed Description

Contains the FilterWindow class which holds the UI to create filters.

Author

Jorge Agüero Zamora

Version

0.1

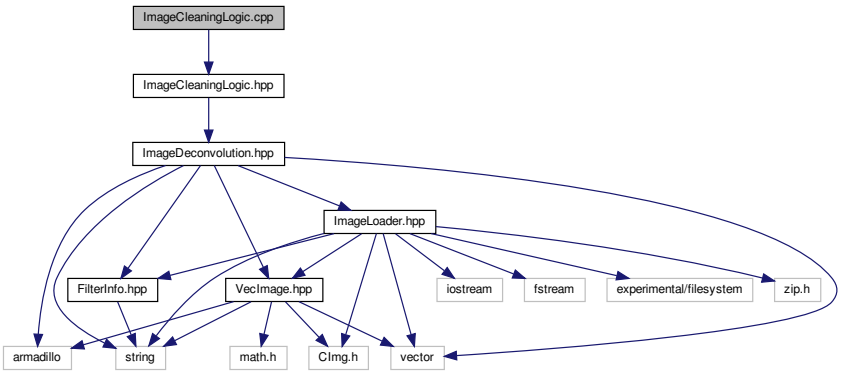
Date

2021-06-13

7.8 ImageCleaningLogic.cpp File Reference

```
#include "ImageCleaningLogic.hpp"
```

Include dependency graph for ImageCleaningLogic.cpp:



Namespaces

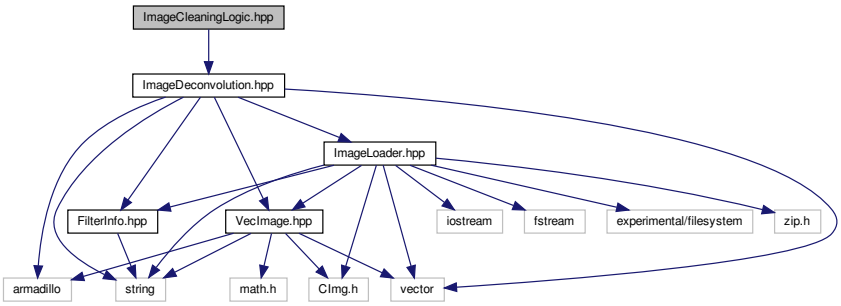
- AppLogic

7.9 ImageCleaningLogic.hpp File Reference

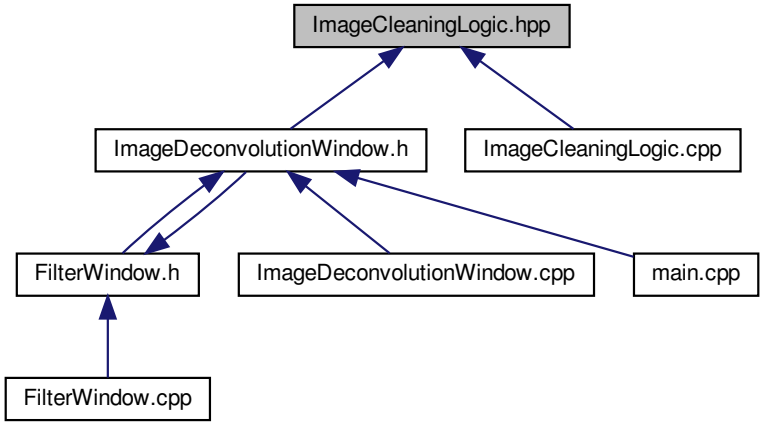
Contains the API to be able to apply filters to images.

```
#include "ImageDeconvolution.hpp"
```

Include dependency graph for ImageCleaningLogic.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AppLogic::ImageCleaningLogic

  API to the ImageDeconvolution class.

Namespaces

- AppLogic

7.9.1 Detailed Description

Contains the API to be able to apply filters to images.

Author

Jorge Agüero Zamora

Version

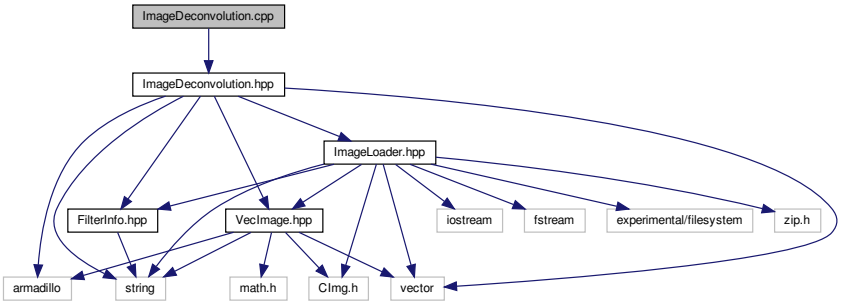
0.1

Date

2021-06-13

7.10 ImageDeconvolution.cpp File Reference

`#include "ImageDeconvolution.hpp"`
Include dependency graph for ImageDeconvolution.cpp:



Namespaces

- AppLogic

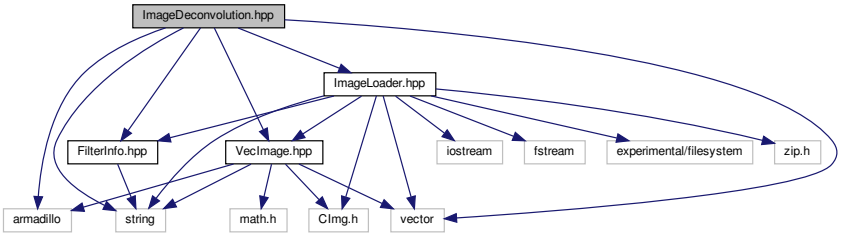
7.11 ImageDeconvolution.hpp File Reference

Contains `ImageDeconvolution` class in charge of applying filters to images.

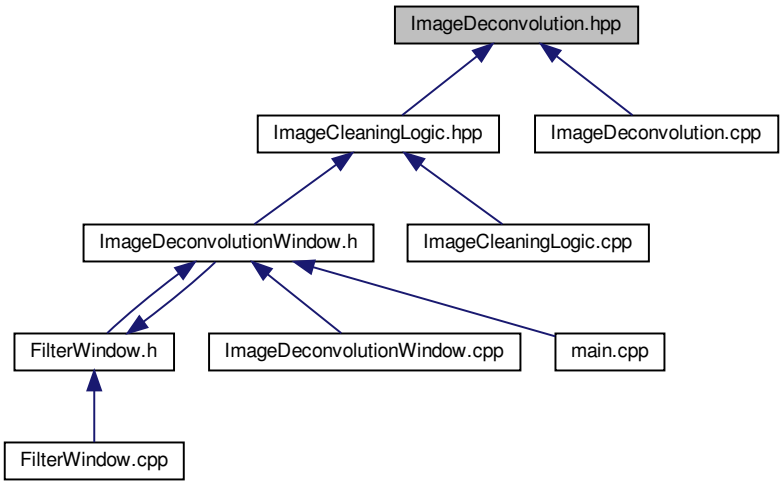
```
#include <vector>
#include <string>
#include <armadillo>
#include "FilterInfo.hpp"
#include "VecImage.hpp"
```

```
#include "ImageLoader.hpp"
```

Include dependency graph for ImageDeconvolution.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AppLogic::ImageDeconvolution

  Allows to apply existing filter to images.

Namespaces

- AppLogic

Macros

- #define FILT_IMAGE_SUFFIX "_filtered.png"

  suffix used in saved filtered image filenames

7.11.1 Detailed Description

Contains `ImageDeconvolution` class in charge of applying filters to images.

Author

Jorge Agüero Zamora

Version

0.1

Date

2021-06-13

7.11.2 Macro Definition Documentation

7.11.2.1 FILT_IMAGE_SUFFIX

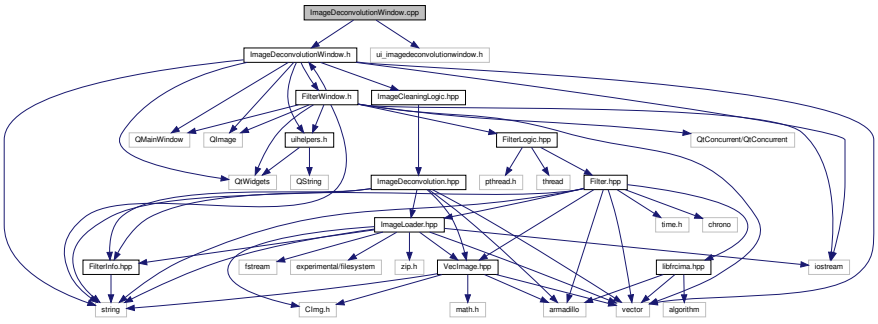
```
#define FILT_IMAGE_SUFFIX "_filtered.png"
```

suffix used in saved filtered image filenames

7.12 ImageDeconvolutionWindow.cpp File Reference

```
#include "ImageDeconvolutionWindow.h"
#include "ui_imagedeconvolutionwindow.h"
```

Include dependency graph for ImageDeconvolutionWindow.cpp:



Namespaces

- UI

Macros

- #define ELIDE_WIDTH 400

7.12.1 Macro Definition Documentation

7.12.1.1 ELIDE_WIDTH

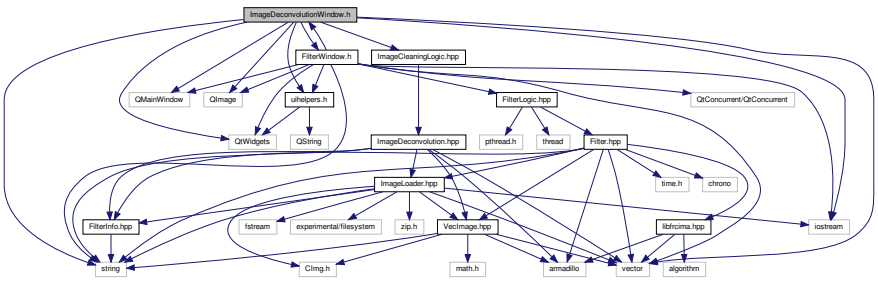
```
#define ELIDE_WIDTH 400
```

7.13 ImageDeconvolutionWindow.h File Reference

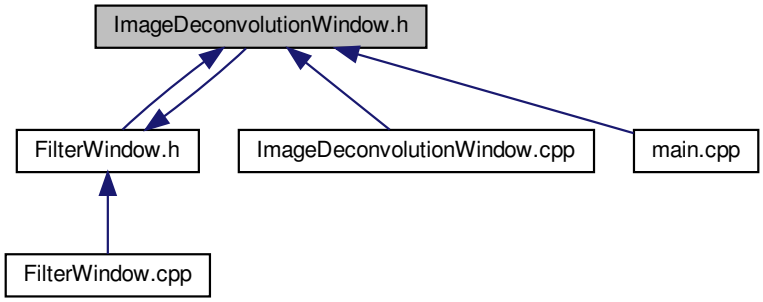
Contains `ImageDeconvolutionWindow` class which holds the UI to apply filters to images.

```
#include <QMainWindow>
#include <QImage>
#include <QtWidgets>
#include <vector>
#include <string>
#include <iostream>
#include "uihelpers.h"
#include "ImageCleaningLogic.hpp"
#include "FilterWindow.h"
```

Include dependency graph for ImageDeconvolutionWindow.h:



This graph shows which files directly or indirectly include this file:



Classes

- class UI::ImageDeconvolutionWindow

  Window to filter images.

Namespaces

- Ui
- UI

7.13.1 Detailed Description

Contains `ImageDeconvolutionWindow` class which holds the UI to apply filters to images.

Author

Jorge Agüero Zamora

Version

0.1

Date

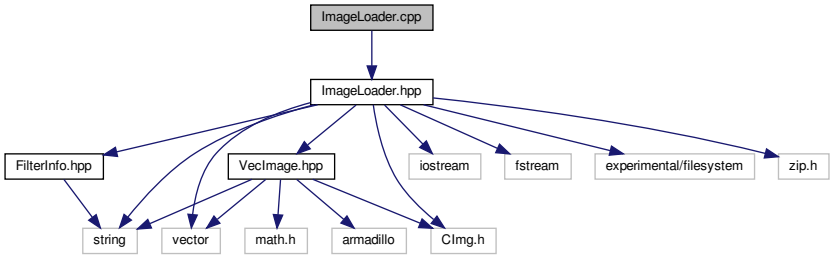
2021-06-13

Copyright

7.14 ImageLoader.cpp File Reference

```
#include "ImageLoader.hpp"
```

Include dependency graph for ImageLoader.cpp:



Namespaces

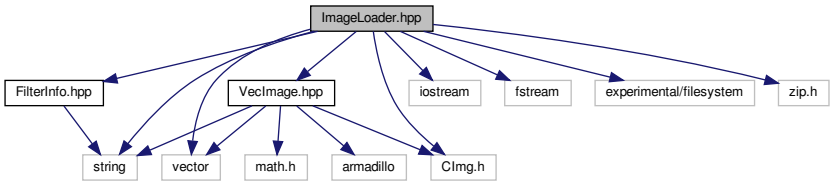
- AppLogic

7.15 ImageLoader.hpp File Reference

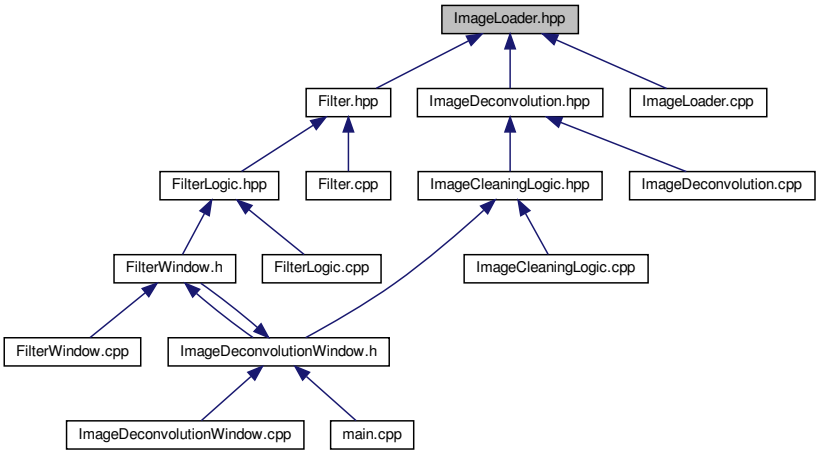
Contains ImageLoader class in charge of loading images and filter information.

```
#include "VecImage.hpp"
#include "FilterInfo.hpp"
#include "CImg.h"
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <experimental/filesystem>
#include <zip.h>
```

Include dependency graph for ImageLoader.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AppLogic::ImageLoader

  Allows to load images and filter information as well as import and export filters.

Namespaces

- AppLogic

Macros

- #define FILTER_INFO_EXTENSION ".finfo"
  file extension for FilterInfo files
- #define FILTER_FILE_EXTENSION ".mat"
  file extension for filter matrix files
- #define TEMP_FILE_NAME "tmpfile"
  name of tempfile to use when extracting from zip
- #define ZIP_FILE_BUFF_SIZE 4096
  size of the buffer to copy data from zip

7.15.1 Detailed Description

Contains ImageLoader class in charge of loading images and filter information.

Author

Jorge Agüero Zamora

Version

0.1

Date

2021-06-13

Copyright

7.15.2 Macro Definition Documentation

7.15.2.1 FILTER_FILE_EXTENSION

```
#define FILTER_FILE_EXTENSION ".mat"
```

file extension for filter matrix files

7.15.2.2 FILTER_INFO_EXTENSION

```
#define FILTER_INFO_EXTENSION ".finfo"
```

file extension for FilterInfo files

7.15.2.3 TEMP_FILE_NAME

```
#define TEMP_FILE_NAME "tmpfile"
```

name of tempfile to use when extracting from zip

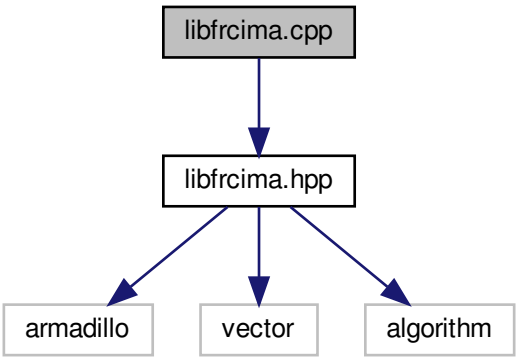
7.15.2.4 ZIP_FILE_BUFF_SIZE

```
#define ZIP_FILE_BUFF_SIZE 4096
```

size of the buffer to copy data from zip

7.16 libfrcima.cpp File Reference

```
#include "libfrcima.hpp"
```
Include dependency graph for libfrcima.cpp:



Namespaces

- frcima

Functions

- void frcima::calculate_error (mat &A, const mat &C, const mat &X)

  calculates the error from the calculated filter
- bool frcima::low_rank_approx (const mat &A, const size_t r, mat &B, mat &C)

  calculates low rank matrix approximation using the modified bilateral random projections (MBRP) method.
- bool frcima::low_rank (const mat &A, const size_t r, mat &B, mat &C)

  calculates low rank matrix using SVD method
- bool frcima::generate_training_matrices (const vector< vector< double >> &X, const vector< vector< double >> &C, mat &X_tr, mat &C_tr)

  generates training matrices from vectors
- bool frcima::pseudo_inverse_tpm (const mat &A, mat &Ap)

  Calculates pseudo inverse matrix using the Tensor Product Matrix (TPM) method.
- mat frcima::rcima (const vector< vector< double >> &t_data_x, const vector< vector< double >> &t_data_c, const size_t rank)

  calculates rank constrained inverse matrix approximation using RCIMA method
- mat frcima::rcima (const mat &X, const mat &C, const size_t rank)

  calculates rank constrained inverse matrix approximation using RCIMA method
- mat frcima::fast_rcima (const vector< vector< double >> &t_data_x, const vector< vector< double >> &t_data_c, const size_t rank)

  calculates rank constrained inverse matrix approximation using fast-RCIMA algorithm
- mat frcima::fast_rcima (const mat &X, const mat &C, const size_t rank)

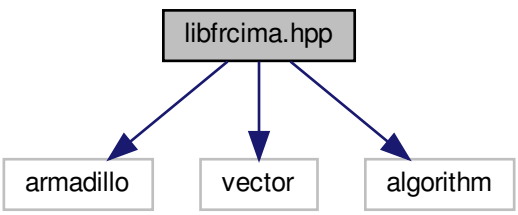
  calculates rank constrained inverse matrix approximation using fast-RCIMA algorithm

7.17 libfrcima.hpp File Reference

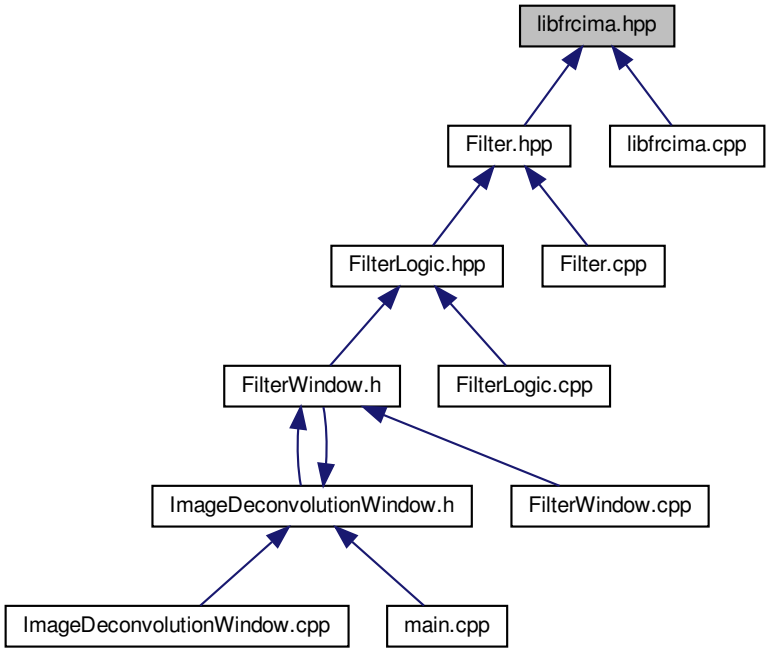
Contains the RCIMA and fast-RCIMA methods.

```
#include <armadillo>
#include <vector>
#include <algorithm>
```

Include dependency graph for libfrcima.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- frcima

Macros

- #define LOW_RANK_APPROX_ITERATIONS 3

  amount of iteration to approximate low rank matrix

Functions

- mat frcima::rcima (const vector< vector< double >> &t_data_x, const vector< vector< double >> &t_data_c, const size_t rank)

  calculates rank constrained inverse matrix approximation using RCIMA method
- mat frcima::rcima (const mat &X, const mat &C, const size_t rank)

  calculates rank constrained inverse matrix approximation using RCIMA method
- mat frcima::fast_rcima (const vector< vector< double >> &t_data_x, const vector< vector< double >> &t_data_c, const size_t rank)

  calculates rank constrained inverse matrix approximation using fast-RCIMA algorithm
- mat frcima::fast_rcima (const mat &X, const mat &C, const size_t rank)

  calculates rank constrained inverse matrix approximation using fast-RCIMA algorithm

Variables

- double frcima::_calculated_error
  global variable to hold calculated error

7.17.1 Detailed Description

Contains the RCIMA and fast-RCIMA methods.

Author

Jorge Agüero Zamora

Version

0.1

Date

2021-06-14

Copyright

Copyright (c) 2021

7.17.2 Macro Definition Documentation

7.17.2.1 LOW_RANK_APPROX_ITERATIONS

```
#define LOW_RANK_APPROX_ITERATIONS 3
```

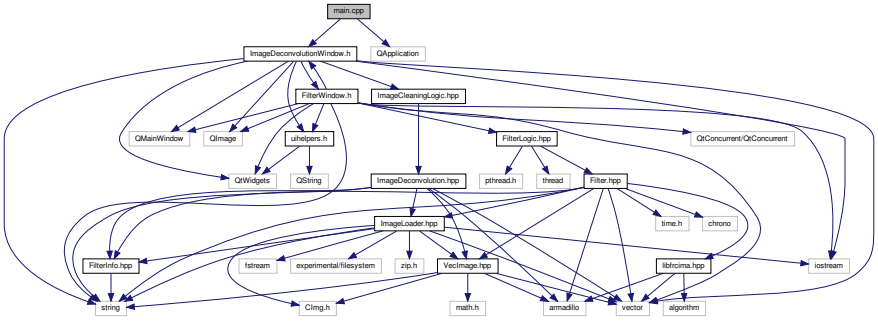
amount of iteration to approximate low rank matrix

7.18 main.cpp File Reference

main entry point for application

```
#include "ImageDeconvolutionWindow.h"
#include <QApplication>
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char ∗argv[ ])

7.18.1 Detailed Description

main entry point for application

Author

Jorge Agüero Zamora

Version

0.1

Date

2021-06-14

7.18.2 Function Documentation

7.18.2.1 main()

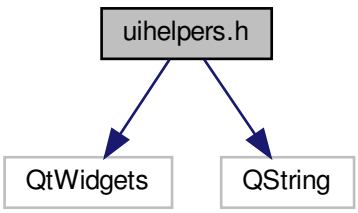
```
int main (
            int argc,
            char * argv[] )
```

7.19 uihelpers.h File Reference

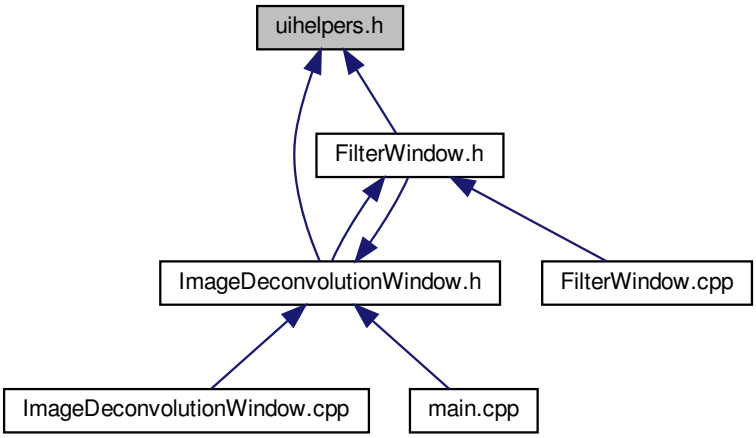
Contains helper functions for the Window classes.

```
#include <QtWidgets>
#include <QString>
```

Include dependency graph for uihelpers.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- uihelp

Macros

- #define ELEMENT_NUMBER_MILLION_CUTOFF 1000000

  one million constant
- #define ELEMENT_NUMBER_THOUSAND_CUTOFF 1000

  on thousand constant
- #define LONGER_MESSAGE_TIME 3000

  time for status messages to appear in window
- #define STANDARD_MESSAGE_TIME 2000

  time for status messages to appear in window

Enumerations

- enum uihelp::file_dialog_type_t { uihelp::FD_IMAGE, uihelp::FD_ZIP }

  enum with file dialog type

Functions

- static QString uihelp::allSupportedFormatsString (QStringList mimeTypeFilters)

  constructs string with all supported MIME types
- static void uihelp::initializeImageFileDialog (QFileDialog &dialog, QFileDialog::AcceptMode acceptMode, file_dialog_type_t file_dialog_type=file_dialog_type_t::FD_IMAGE, QFileDialog::FileMode filemode=QFile↩Dialog::FileMode::DirectoryOnly)

  creates a file dialog window

- static void uihelp::hideListItems (QListWidget *list)
  hides all items in a QListWidget
- static void uihelp::filterListItems (QListWidget *list, QString filter_string)
  filters the list contents that contain the filter string
- static QString uihelp::generateSizeMessage (int rows, int cols)
  generates string with size information

7.19.1 Detailed Description

Contains helper functions for the Window classes.

Author

Jorge Agüero Zamora

Version

0.1

Date

2021-06-14

7.19.2 Macro Definition Documentation

7.19.2.1 ELEMENT_NUMBER_MILLION_CUTOFF

```
#define ELEMENT_NUMBER_MILLION_CUTOFF 1000000
```

one million constant

7.19.2.2 ELEMENT_NUMBER_THOUSAND_CUTOFF

```
#define ELEMENT_NUMBER_THOUSAND_CUTOFF 1000
```

on thousand constant

7.19.2.3 LONGER_MESSAGE_TIME

```
#define LONGER_MESSAGE_TIME 3000
```

time for status messages to appear in window

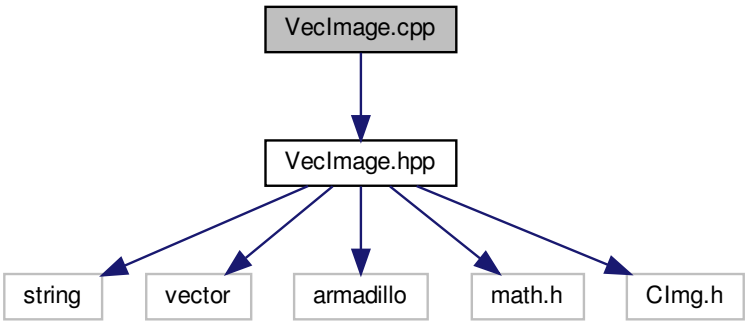
7.19.2.4 STANDARD_MESSAGE_TIME

```
#define STANDARD_MESSAGE_TIME 2000
```

time for status messages to appear in window

7.20 VecImage.cpp File Reference

```
#include "VecImage.hpp"
```
Include dependency graph for VecImage.cpp:



Namespaces

- AppLogic

Macros

- #define VECIMG_NORM 255

7.20.1 Macro Definition Documentation

7.20.1.1 VECIMG_NORM

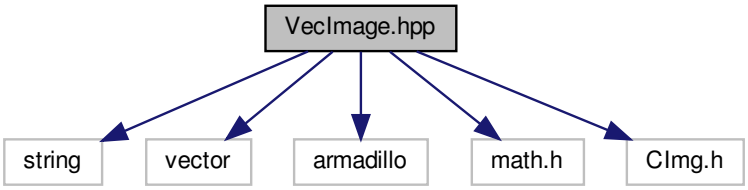
```
#define VECIMG_NORM 255
```

7.21 VecImage.hpp File Reference

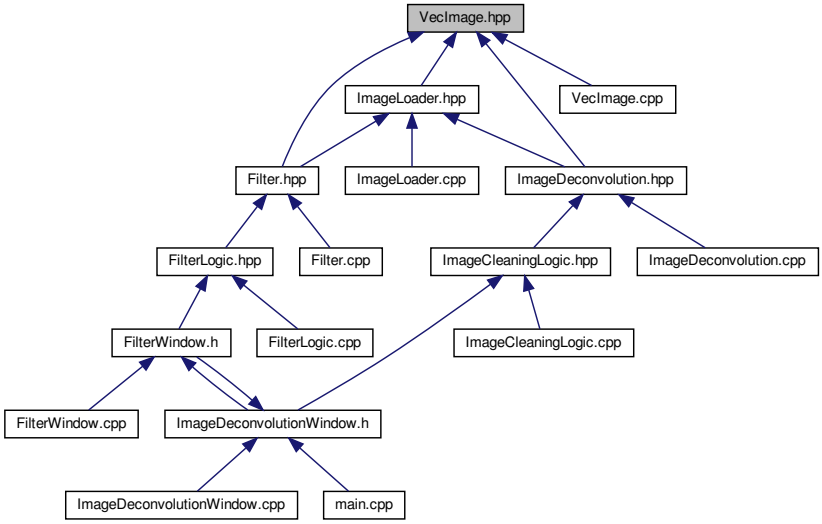
Contains the `VecImage` class used to manipulate images.

```
#include <string>
#include <vector>
#include <armadillo>
#include <math.h>
#include "CImg.h"
```

Include dependency graph for VecImage.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AppLogic::VecImage

  Class used for image manipulation.

Namespaces

- AppLogic

Macros

- #define cimg_use_jpeg
- #define cimg_use_png
- #define cimg_display 0

Enumerations

- enum AppLogic::noise_type_t { AppLogic::GAUSSIAN, AppLogic::SALT_PEPPER, AppLogic::UNIFORM, AppLogic::POISSON, AppLogic::RICIAN }

  enum with the different type of noises that can be applied to a `VecImage`

7.21.1 Detailed Description

Contains the `VecImage` class used to manipulate images.

Author

Jorge Agüero Zamora

Version

0.1

Date

2021-06-13

7.21.2 Macro Definition Documentation

7.21.2.1 cimg_display

```
#define cimg_display 0
```

7.21.2.2 cimg_use_jpeg

```
#define cimg_use_jpeg
```

7.21.2.3 cimg_use_png

```
#define cimg_use_png
```

Index