

# FSP\_ImageDeconvolution

0.1

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Namespace Documentation</b>	<b>9</b>
5.1	AppLogic Namespace Reference . . . . .	9
5.1.1	Enumeration Type Documentation . . . . .	10
5.1.1.1	calc_method_t . . . . .	10
5.1.1.2	noise_type_t . . . . .	10
5.1.2	Function Documentation . . . . .	10
5.1.2.1	operator<<() . . . . .	10
5.1.2.2	operator>>() . . . . .	11
5.2	frcima Namespace Reference . . . . .	11
5.2.1	Function Documentation . . . . .	12
5.2.1.1	calculate_error() . . . . .	12
5.2.1.2	fast_rcima() [1/2] . . . . .	12
5.2.1.3	fast_rcima() [2/2] . . . . .	13

5.2.1.4	<a href="#">generate_training_matrices()</a>	13
5.2.1.5	<a href="#">low_rank()</a>	14
5.2.1.6	<a href="#">low_rank_approx()</a>	14
5.2.1.7	<a href="#">pseudo_inverse_tpm()</a>	15
5.2.1.8	<a href="#">rcima()</a> [1/2]	15
5.2.1.9	<a href="#">rcima()</a> [2/2]	15
5.2.2	<a href="#">Variable Documentation</a>	16
5.2.2.1	<a href="#">_calculated_error</a>	16
5.3	<a href="#">UI Namespace Reference</a>	16
5.3.1	<a href="#">Variable Documentation</a>	16
5.3.1.1	<a href="#">notAllowedChars</a>	16
5.3.1.2	<a href="#">notAllowedSubStrings</a>	17
5.4	<a href="#">Ui Namespace Reference</a>	17
5.5	<a href="#">uihelp Namespace Reference</a>	17
5.5.1	<a href="#">Enumeration Type Documentation</a>	17
5.5.1.1	<a href="#">file_dialog_type_t</a>	17
5.5.2	<a href="#">Function Documentation</a>	18
5.5.2.1	<a href="#">allSupportedFormatsString()</a>	18
5.5.2.2	<a href="#">filterListItems()</a>	18
5.5.2.3	<a href="#">generateSizeMessage()</a>	18
5.5.2.4	<a href="#">hideListItems()</a>	19
5.5.2.5	<a href="#">initializeImageFileDialog()</a>	19

<b>6 Class Documentation</b>	<b>21</b>
6.1 AppLogic::Filter Class Reference	21
6.1.1 Detailed Description	24
6.1.2 Constructor & Destructor Documentation	24
6.1.2.1 Filter()	24
6.1.3 Member Function Documentation	24
6.1.3.1 applyNoiseToImage()	24
6.1.3.2 applyNoiseToWorkingImages()	25
6.1.3.3 calculateFilter()	25
6.1.3.4 cancelSaveDirtyImages()	25
6.1.3.5 clearWorkingImages()	25
6.1.3.6 getCalculationMethodName()	26
6.1.3.7 getFilterInfo()	26
6.1.3.8 getFmatrix()	26
6.1.3.9 getImagePath()	26
6.1.3.10 getLoadedImage()	27
6.1.3.11 getWorkingImagesMat()	27
6.1.3.12 loadFmatrixFromFile()	27
6.1.3.13 loadImagesFromFolder()	28
6.1.3.14 loadImagesFromZip()	28
6.1.3.15 loadSingleImage()	29
6.1.3.16 saveAllDirtyImages()	29
6.1.3.17 saveDirtyImage()	29
6.1.3.18 saveFilterInfo()	30
6.1.3.19 saveToFile()	30
6.1.3.20 setFilterName()	31
6.1.4 Member Data Documentation	31
6.1.4.1 calc_info	31
6.1.4.2 F_matrix	31
6.1.4.3 image_set	31

6.1.4.4	<a href="#">is_canceled</a>	31
6.1.4.5	<a href="#">last_used_noise_type</a>	32
6.1.4.6	<a href="#">last_used_noise_value</a>	32
6.1.4.7	<a href="#">loaded_file_names</a>	32
6.1.4.8	<a href="#">loaded_file_paths</a>	32
6.1.4.9	<a href="#">working_images</a>	32
6.2	<a href="#">AppLogic::FilterInfo Struct Reference</a>	33
6.2.1	<a href="#">Detailed Description</a>	34
6.2.2	<a href="#">Member Data Documentation</a>	34
6.2.2.1	<a href="#">calc_time_seconds</a>	34
6.2.2.2	<a href="#">calculation_method</a>	34
6.2.2.3	<a href="#">f_matrix_num_col</a>	34
6.2.2.4	<a href="#">f_matrix_num_row</a>	34
6.2.2.5	<a href="#">file_image_source</a>	35
6.2.2.6	<a href="#">filter_name</a>	35
6.2.2.7	<a href="#">image_calc_amount</a>	35
6.2.2.8	<a href="#">noise_type</a>	35
6.2.2.9	<a href="#">noise_value</a>	35
6.2.2.10	<a href="#">rank</a>	35
6.3	<a href="#">AppLogic::FilterLogic Class Reference</a>	36
6.3.1	<a href="#">Detailed Description</a>	38
6.3.2	<a href="#">Constructor &amp; Destructor Documentation</a>	38
6.3.2.1	<a href="#">FilterLogic()</a>	38
6.3.3	<a href="#">Member Function Documentation</a>	38
6.3.3.1	<a href="#">addImageToWorkingImages()</a>	38
6.3.3.2	<a href="#">applyNoiseToImage()</a>	38
6.3.3.3	<a href="#">cancelFilterCreation()</a>	39
6.3.3.4	<a href="#">cancelSaveDirtyImages()</a>	39
6.3.3.5	<a href="#">clearWorkingImages()</a>	39
6.3.3.6	<a href="#">createFilter()</a>	39

6.3.3.7	<a href="#">getFilterInfo()</a>	40
6.3.3.8	<a href="#">getImagePath()</a>	40
6.3.3.9	<a href="#">getLoadedImage()</a>	40
6.3.3.10	<a href="#">loadImagesFromFolder()</a>	41
6.3.3.11	<a href="#">loadImagesFromZip()</a>	41
6.3.3.12	<a href="#">saveAllDirtyImages()</a>	42
6.3.3.13	<a href="#">saveDirtyImage()</a>	42
6.3.3.14	<a href="#">setNoise()</a>	42
6.3.3.15	<a href="#">setNoiseValue()</a>	43
6.3.4	<a href="#">Member Data Documentation</a>	43
6.3.4.1	<a href="#">filter_calc_thread</a>	43
6.3.4.2	<a href="#">is_canceled</a>	43
6.3.4.3	<a href="#">last_noise</a>	43
6.3.4.4	<a href="#">last_noise_value</a>	44
6.3.4.5	<a href="#">new_filter</a>	44
6.4	<a href="#">UI::FilterWindow Class Reference</a>	44
6.4.1	<a href="#">Detailed Description</a>	48
6.4.2	<a href="#">Constructor &amp; Destructor Documentation</a>	48
6.4.2.1	<a href="#">FilterWindow()</a>	48
6.4.2.2	<a href="#">~FilterWindow()</a>	49
6.4.3	<a href="#">Member Function Documentation</a>	49
6.4.3.1	<a href="#">createFilterInfoDialog()</a>	49
6.4.3.2	<a href="#">getSelectedNoiseValue()</a>	49
6.4.3.3	<a href="#">loadImageList()</a>	49
6.4.3.4	<a href="#">loadImagesFromFile()</a>	50
6.4.3.5	<a href="#">loadImagesFromFolder()</a>	50
6.4.3.6	<a href="#">loadImagesFromZip()</a>	50
6.4.3.7	<a href="#">on_actionFilter_image_triggered</a>	51
6.4.3.8	<a href="#">on_actionLoadImageFromZip_triggered</a>	51
6.4.3.9	<a href="#">on_btn_load_folder_clicked</a>	51

6.4.3.10	<a href="#">on_btn_load_single_image_clicked</a>	51
6.4.3.11	<a href="#">on_btn_save_all_dirtyimg_clicked</a>	51
6.4.3.12	<a href="#">on_btn_save_dirtyimg_clicked</a>	51
6.4.3.13	<a href="#">on_calc_filter_btn_clicked</a>	52
6.4.3.14	<a href="#">on_filter_name_input_editingFinished</a>	52
6.4.3.15	<a href="#">on_filterCalculationCanceled</a>	52
6.4.3.16	<a href="#">on_filterCalculationFinished</a>	52
6.4.3.17	<a href="#">on_lineEdit_filter_loaded_images_textChanged</a>	52
6.4.3.18	<a href="#">on_listWidget_loaded_images_currentRowChanged</a>	52
6.4.3.19	<a href="#">on_loadImagesCanceled</a>	52
6.4.3.20	<a href="#">on_loadImagesFinished</a>	53
6.4.3.21	<a href="#">on_next_loaded_image_btn_clicked</a>	53
6.4.3.22	<a href="#">on_noise_selection_group_currentChanged</a>	53
6.4.3.23	<a href="#">on_previous_loaded_image_btn_clicked</a>	53
6.4.3.24	<a href="#">on_saveAllDirtyImagesCanceled</a>	53
6.4.3.25	<a href="#">on_saveAllDirtyImagesFinished</a>	53
6.4.3.26	<a href="#">on_slider_gaussiannoise_valueChanged</a>	53
6.4.3.27	<a href="#">on_slider_riciannoise_valueChanged</a>	54
6.4.3.28	<a href="#">on_slider_snpnoise_valueChanged</a>	54
6.4.3.29	<a href="#">on_slider_uniformnoise_valueChanged</a>	54
6.4.3.30	<a href="#">saveAllDirtyImages()</a>	54
6.4.3.31	<a href="#">saveDirtyImage()</a>	54
6.4.3.32	<a href="#">setDirtyImage()</a>	55
6.4.3.33	<a href="#">setFilterNameInputError()</a>	55
6.4.3.34	<a href="#">setLoadedImage()</a>	55
6.4.3.35	<a href="#">validateNameInput()</a>	56
6.4.4	<a href="#">Member Data Documentation</a>	56
6.4.4.1	<a href="#">bool_future_watcher</a>	56
6.4.4.2	<a href="#">dirty_image</a>	56
6.4.4.3	<a href="#">filter_logic</a>	56



6.4.4.4	<a href="#">future_watcher_calc_filter</a>	56
6.4.4.5	<a href="#">future_watcher_load_images</a>	57
6.4.4.6	<a href="#">image</a>	57
6.4.4.7	<a href="#">label_dirty_image</a>	57
6.4.4.8	<a href="#">label_loaded_image</a>	57
6.4.4.9	<a href="#">progress_dialog_calc_filter</a>	57
6.4.4.10	<a href="#">selected_noise_type</a>	57
6.4.4.11	<a href="#">selected_noise_value</a>	58
6.4.4.12	<a href="#">ui</a>	58
6.5	<a href="#">AppLogic::ImageCleaningLogic Class Reference</a>	58
6.5.1	<a href="#">Detailed Description</a>	60
6.5.2	<a href="#">Member Function Documentation</a>	60
6.5.2.1	<a href="#">addImageToWorkingImages()</a>	60
6.5.2.2	<a href="#">applyFilterToImage()</a>	61
6.5.2.3	<a href="#">cancelFilterAllImages()</a>	61
6.5.2.4	<a href="#">clearWorkingImages()</a>	61
6.5.2.5	<a href="#">deleteFilter()</a>	61
6.5.2.6	<a href="#">getFilterInfo()</a>	62
6.5.2.7	<a href="#">getFilterNames()</a>	62
6.5.2.8	<a href="#">getImagePath()</a>	62
6.5.2.9	<a href="#">getLoadedImage()</a>	63
6.5.2.10	<a href="#">loadFilter()</a>	63
6.5.2.11	<a href="#">loadImagesFromFolder()</a>	63
6.5.2.12	<a href="#">loadImagesFromZip()</a>	64
6.5.2.13	<a href="#">saveAllFilteredImages()</a>	64
6.5.2.14	<a href="#">saveFilteredImage()</a>	65
6.5.3	<a href="#">Member Data Documentation</a>	65
6.5.3.1	<a href="#">deconv</a>	65
6.5.3.2	<a href="#">filters</a>	65
6.6	<a href="#">AppLogic::ImageDeconvolution Class Reference</a>	66

6.6.1	Detailed Description	67
6.6.2	Constructor & Destructor Documentation	67
6.6.2.1	ImageDeconvolution()	67
6.6.2.2	~ImageDeconvolution()	68
6.6.3	Member Function Documentation	68
6.6.3.1	applyFilterToImage()	68
6.6.3.2	cancelFilterAllImages()	68
6.6.3.3	clearWorkingImages()	68
6.6.3.4	deleteFilter()	68
6.6.3.5	getImagePath()	69
6.6.3.6	getLoadedFilters()	69
6.6.3.7	getLoadedImage()	69
6.6.3.8	loadExistingFilters()	70
6.6.3.9	loadFilter()	70
6.6.3.10	loadImagesFromFolder()	70
6.6.3.11	loadImagesFromZip()	72
6.6.3.12	loadSingleImage()	72
6.6.3.13	saveAllFilteredImages()	73
6.6.3.14	saveFilteredImage()	73
6.6.4	Member Data Documentation	73
6.6.4.1	F_matrix	73
6.6.4.2	is_canceled	74
6.6.4.3	loaded_file_names	74
6.6.4.4	loaded_file_paths	74
6.6.4.5	loaded_filters	74
6.6.4.6	working_images	74
6.7	UI::ImageDeconvolutionWindow Class Reference	75
6.7.1	Detailed Description	78
6.7.2	Constructor & Destructor Documentation	78
6.7.2.1	ImageDeconvolutionWindow()	78

6.7.2.2	<a href="#">~ImageDeconvolutionWindow()</a> . . . . .	79
6.7.3	Member Function Documentation . . . . .	79
6.7.3.1	<a href="#">displayFilterInfo()</a> . . . . .	79
6.7.3.2	<a href="#">exportFilter()</a> . . . . .	79
6.7.3.3	<a href="#">importFilter()</a> . . . . .	79
6.7.3.4	<a href="#">loadImageList()</a> . . . . .	80
6.7.3.5	<a href="#">loadImagesFromFile()</a> . . . . .	80
6.7.3.6	<a href="#">loadImagesFromFolder()</a> . . . . .	80
6.7.3.7	<a href="#">loadImagesFromZip()</a> . . . . .	81
6.7.3.8	<a href="#">on_actionCreate_filter_triggered</a> . . . . .	81
6.7.3.9	<a href="#">on_actionLoadImageFromZip_triggered</a> . . . . .	81
6.7.3.10	<a href="#">on_btn_load_folder_clicked</a> . . . . .	81
6.7.3.11	<a href="#">on_btn_load_image_clicked</a> . . . . .	81
6.7.3.12	<a href="#">on_comboBox_filter_select_currentIndexChanged</a> . . . . .	82
6.7.3.13	<a href="#">on_exportFilterCanceled</a> . . . . .	82
6.7.3.14	<a href="#">on_exportFilterFinished</a> . . . . .	82
6.7.3.15	<a href="#">on_filterAllImagesCanceled</a> . . . . .	82
6.7.3.16	<a href="#">on_filterAllImagesFinished</a> . . . . .	82
6.7.3.17	<a href="#">on_filterLoadingFinished</a> . . . . .	82
6.7.3.18	<a href="#">on_importFilterCanceled</a> . . . . .	82
6.7.3.19	<a href="#">on_importFilterFinished</a> . . . . .	83
6.7.3.20	<a href="#">on_lineEdit_filter_loaded_images_textChanged</a> . . . . .	83
6.7.3.21	<a href="#">on_listWidget_loaded_images_currentRowChanged</a> . . . . .	83
6.7.3.22	<a href="#">on_loadImagesCanceled</a> . . . . .	83
6.7.3.23	<a href="#">on_loadImagesFinished</a> . . . . .	83
6.7.3.24	<a href="#">on_next_loaded_image_btn_clicked</a> . . . . .	83
6.7.3.25	<a href="#">on_previous_loaded_image_btn_clicked</a> . . . . .	83
6.7.3.26	<a href="#">on_pushButton_deletefilter_clicked</a> . . . . .	84
6.7.3.27	<a href="#">on_pushButton_exportfilter_clicked</a> . . . . .	84
6.7.3.28	<a href="#">on_pushButton_filterall_save_clicked</a> . . . . .	84

6.7.3.29	<a href="#">on_pushButton_importfilter_clicked</a>	84
6.7.3.30	<a href="#">on_pushButton_save_filterimage_clicked</a>	84
6.7.3.31	<a href="#">saveAllFilteredImages()</a>	84
6.7.3.32	<a href="#">saveFilteredImage()</a>	85
6.7.3.33	<a href="#">setFilteredImage()</a>	85
6.7.3.34	<a href="#">setLoadedImage()</a>	85
6.7.3.35	<a href="#">updateFilterList()</a>	86
6.7.4	<a href="#">Member Data Documentation</a>	86
6.7.4.1	<a href="#">bool_future_watcher</a>	86
6.7.4.2	<a href="#">filtered_image</a>	86
6.7.4.3	<a href="#">future_watcher_load_images</a>	86
6.7.4.4	<a href="#">image</a>	86
6.7.4.5	<a href="#">img_cleaning</a>	87
6.7.4.6	<a href="#">is_filter_loaded</a>	87
6.7.4.7	<a href="#">label_filtered_image</a>	87
6.7.4.8	<a href="#">label_loaded_image</a>	87
6.7.4.9	<a href="#">loadfilter_future_watcher</a>	87
6.7.4.10	<a href="#">progress_dialog_filter</a>	87
6.7.4.11	<a href="#">ui</a>	88
6.8	<a href="#">AppLogic::ImageLoader Class Reference</a>	88
6.8.1	<a href="#">Detailed Description</a>	90
6.8.2	<a href="#">Member Function Documentation</a>	90
6.8.2.1	<a href="#">addImageFileInfo()</a>	90
6.8.2.2	<a href="#">cancelOperation()</a>	91
6.8.2.3	<a href="#">clearLoadedData()</a>	91
6.8.2.4	<a href="#">createExportZip()</a>	91
6.8.2.5	<a href="#">createFileFromZipFile()</a>	92
6.8.2.6	<a href="#">deleteFilter()</a>	92
6.8.2.7	<a href="#">endsWith()</a>	92
6.8.2.8	<a href="#">existsNameFilter()</a>	93

6.8.2.9	<a href="#">getLoadedImageNames()</a>	93
6.8.2.10	<a href="#">getLoadedImagePaths()</a>	93
6.8.2.11	<a href="#">getNotLoadedImagePaths()</a>	94
6.8.2.12	<a href="#">getSourceDirectory()</a>	94
6.8.2.13	<a href="#">importFilterFromZip()</a>	94
6.8.2.14	<a href="#">isSupportedImageFormat()</a>	94
6.8.2.15	<a href="#">loadFilterInfo()</a>	95
6.8.2.16	<a href="#">loadImagesFromFolder()</a>	95
6.8.2.17	<a href="#">loadImagesFromZip()</a>	95
6.8.2.18	<a href="#">loadSingleImage()</a>	97
6.8.2.19	<a href="#">stripExtensionFromFilename()</a>	97
6.8.3	<a href="#">Member Data Documentation</a>	97
6.8.3.1	<a href="#">_is_canceled</a>	98
6.8.3.2	<a href="#">FILTER_SAVE_LOCATION</a>	98
6.8.3.3	<a href="#">loaded_file_names</a>	98
6.8.3.4	<a href="#">loaded_file_paths</a>	98
6.8.3.5	<a href="#">not_loaded_files</a>	98
6.8.3.6	<a href="#">source_directory</a>	98
6.9	<a href="#">AppLogic::VecImage Class Reference</a>	99
6.9.1	<a href="#">Detailed Description</a>	100
6.9.2	<a href="#">Constructor &amp; Destructor Documentation</a>	101
6.9.2.1	<a href="#">VecImage() [1/6]</a>	101
6.9.2.2	<a href="#">VecImage() [2/6]</a>	101
6.9.2.3	<a href="#">VecImage() [3/6]</a>	101
6.9.2.4	<a href="#">VecImage() [4/6]</a>	102
6.9.2.5	<a href="#">VecImage() [5/6]</a>	102
6.9.2.6	<a href="#">VecImage() [6/6]</a>	102
6.9.3	<a href="#">Member Function Documentation</a>	102
6.9.3.1	<a href="#">applyNoise()</a>	103
6.9.3.2	<a href="#">getDoubleData()</a>	103

6.9.3.3	<a href="#">getMatDouble()</a>	103
6.9.3.4	<a href="#">getNoiseName()</a>	103
6.9.3.5	<a href="#">getRawImageData()</a>	104
6.9.3.6	<a href="#">getSourceFileLocation()</a>	104
6.9.3.7	<a href="#">getVecDoubleData()</a>	104
6.9.3.8	<a href="#">numCols()</a>	105
6.9.3.9	<a href="#">numRows()</a>	105
6.9.3.10	<a href="#">operator=()</a>	105
6.9.3.11	<a href="#">save()</a>	105
6.9.3.12	<a href="#">setImgDataMatDouble()</a>	106
6.9.4	<a href="#">Member Data Documentation</a>	106
6.9.4.1	<a href="#">cols</a>	106
6.9.4.2	<a href="#">data</a>	106
6.9.4.3	<a href="#">rows</a>	106
6.9.4.4	<a href="#">source_file_location</a>	107
<b>7</b>	<b><a href="#">File Documentation</a></b>	<b>109</b>
7.1	<a href="#">Filter.cpp File Reference</a>	109
7.2	<a href="#">Filter.hpp File Reference</a>	109
7.2.1	<a href="#">Detailed Description</a>	111
7.2.2	<a href="#">Macro Definition Documentation</a>	111
7.2.2.1	<a href="#">DIRTY_IMAGE_SUFFIX</a>	111
7.3	<a href="#">FilterInfo.hpp File Reference</a>	111
7.3.1	<a href="#">Detailed Description</a>	112
7.4	<a href="#">FilterLogic.cpp File Reference</a>	113
7.5	<a href="#">FilterLogic.hpp File Reference</a>	113
7.5.1	<a href="#">Detailed Description</a>	114
7.6	<a href="#">FilterWindow.cpp File Reference</a>	115
7.6.1	<a href="#">Macro Definition Documentation</a>	115
7.6.1.1	<a href="#">SLASHES</a>	115
7.6.1.2	<a href="#">WINDOWS_DEVICES</a>	115

7.7	FilterWindow.h File Reference	116
7.7.1	Detailed Description	117
7.8	ImageCleaningLogic.cpp File Reference	117
7.9	ImageCleaningLogic.hpp File Reference	118
7.9.1	Detailed Description	119
7.10	ImageDeconvolution.cpp File Reference	119
7.11	ImageDeconvolution.hpp File Reference	119
7.11.1	Detailed Description	121
7.11.2	Macro Definition Documentation	121
7.11.2.1	FILT_IMAGE_SUFFIX	121
7.12	ImageDeconvolutionWindow.cpp File Reference	121
7.12.1	Macro Definition Documentation	122
7.12.1.1	ELIDE_WIDTH	122
7.13	ImageDeconvolutionWindow.h File Reference	122
7.13.1	Detailed Description	123
7.14	ImageLoader.cpp File Reference	123
7.15	ImageLoader.hpp File Reference	124
7.15.1	Detailed Description	125
7.15.2	Macro Definition Documentation	125
7.15.2.1	FILTER_FILE_EXTENSION	125
7.15.2.2	FILTER_INFO_EXTENSION	126
7.15.2.3	TEMP_FILE_NAME	126
7.15.2.4	ZIP_FILE_BUFF_SIZE	126
7.16	libfrcima.cpp File Reference	126
7.17	libfrcima.hpp File Reference	127
7.17.1	Detailed Description	129
7.17.2	Macro Definition Documentation	129
7.17.2.1	LOW_RANK_APPROX_ITERATIONS	129
7.18	main.cpp File Reference	129
7.18.1	Detailed Description	130

7.18.2	Function Documentation	130
7.18.2.1	main()	130
7.19	uihelpers.h File Reference	130
7.19.1	Detailed Description	132
7.19.2	Macro Definition Documentation	132
7.19.2.1	ELEMENT_NUMBER_MILLION_CUTOFF	132
7.19.2.2	ELEMENT_NUMBER_THOUSAND_CUTOFF	132
7.19.2.3	LONGER_MESSAGE_TIME	132
7.19.2.4	STANDARD_MESSAGE_TIME	133
7.20	VecImage.cpp File Reference	133
7.20.1	Macro Definition Documentation	133
7.20.1.1	VECIMG_NORM	133
7.21	VecImage.hpp File Reference	134
7.21.1	Detailed Description	135
7.21.2	Macro Definition Documentation	135
7.21.2.1	cimg_display	135
7.21.2.2	cimg_use_jpeg	135
7.21.2.3	cimg_use_png	135
<b>Index</b>		<b>137</b>



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">AppLogic</a>	9
<a href="#">frcima</a>	11
<a href="#">UI</a>	16
<a href="#">Ui</a>	17
<a href="#">uihelp</a>	17



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

AppLogic::Filter . . . . .	21
AppLogic::FilterInfo . . . . .	33
AppLogic::FilterLogic . . . . .	36
AppLogic::ImageCleaningLogic . . . . .	58
AppLogic::ImageDeconvolution . . . . .	66
AppLogic::ImageLoader . . . . .	88
QMainWindow	
UI::FilterWindow . . . . .	44
UI::ImageDeconvolutionWindow . . . . .	75
AppLogic::VecImage . . . . .	99



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">AppLogic::Filter</a>	
Allows to create filters based on training sets of images . . . . .	21
<a href="#">AppLogic::FilterInfo</a>	
Struct to hold the the information of the calculated filter . . . . .	33
<a href="#">AppLogic::FilterLogic</a>	
API to the <a href="#">Filter</a> class . . . . .	36
<a href="#">UI::FilterWindow</a>	
Window to create filters . . . . .	44
<a href="#">AppLogic::ImageCleaningLogic</a>	
API to the <a href="#">ImageDeconvolution</a> class . . . . .	58
<a href="#">AppLogic::ImageDeconvolution</a>	
Allows to apply existing filter to images . . . . .	66
<a href="#">UI::ImageDeconvolutionWindow</a>	
Window to filter images . . . . .	75
<a href="#">AppLogic::ImageLoader</a>	
Allows to load images and filter information as well as import and export filters . . . . .	88
<a href="#">AppLogic::VecImage</a>	
Class used for image manipulation . . . . .	99



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Filter.cpp</a>	109
<a href="#">Filter.hpp</a>	
Contains <code>Filter</code> class in charge of creating filters	109
<a href="#">FilterInfo.hpp</a>	
Contains struct to store filter calculation information and the members to save this information into files	111
<a href="#">FilterLogic.cpp</a>	113
<a href="#">FilterLogic.hpp</a>	
Contains the API for filter creation	113
<a href="#">FilterWindow.cpp</a>	115
<a href="#">FilterWindow.h</a>	
Contains the <code>FilterWindow</code> class which holds the <code>UI</code> to create filters	116
<a href="#">ImageCleaningLogic.cpp</a>	117
<a href="#">ImageCleaningLogic.hpp</a>	
Contains the API to be able to apply filters to images	118
<a href="#">ImageDeconvolution.cpp</a>	119
<a href="#">ImageDeconvolution.hpp</a>	
Contains <code>ImageDeconvolution</code> class in charge of applying filters to images	119
<a href="#">ImageDeconvolutionWindow.cpp</a>	121
<a href="#">ImageDeconvolutionWindow.h</a>	
Contains <code>ImageDeconvolutionWindow</code> class which holds the <code>UI</code> to apply filters to images	122
<a href="#">ImageLoader.cpp</a>	123
<a href="#">ImageLoader.hpp</a>	
Contains <code>ImageLoader</code> class in charge of loading images and filter information	124
<a href="#">librcima.cpp</a>	126
<a href="#">librcima.hpp</a>	
Contains the RCIMA and fast-RCIMA methods	127
<a href="#">main.cpp</a>	
Main entry point for application	129
<a href="#">uihelpers.h</a>	
Contains helper functions for the Window classes	130
<a href="#">VecImage.cpp</a>	133
<a href="#">VecImage.hpp</a>	
Contains the <code>VecImage</code> class used to manipulate images	134





## 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

<i>stream</i>	where the filter information is written
<i>data</i>	filter information to write to stream

## Returns

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

## 5.1.2.2 operator&gt;&gt;()

```
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

<i>stream</i>	stream to read filter information from
<i>data</i>	<a href="#">FilterInfo</a> 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

<i>A</i>	Calculated filter we want to get the error from
<i>C</i>	Training matrix with vectorized noisy images
<i>X</i>	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

<i>t_data_x</i>	the original source images. The training data is a vector of vectors containing a vectorized form of an image
<i>t_data_c</i>	the images with noise. The training data is a vector of vectors containing a vectorized form of an image
<i>rank</i>	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

$t\_data \leftrightarrow \_x$	the original source images. The training data is an arma::mat where each column is a vectorized form of a training image
$t\_data \leftrightarrow \_c$	the images with noise. The training data is an arma::mat where each column is a vectorized form of a training image
<i>rank</i>	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

<i>X</i>	input training vector
<i>C</i>	input training vector
$X \leftrightarrow \_tr$	ouput training matrix
$C \leftrightarrow \_tr$	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

<i>A</i>	input matrix to calculate low rank
<i>r</i>	desired rank, must be less than the minimum between the number of columns and rows of A
<i>B</i>	output left bilateral matrix
<i>C</i>	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

<i>A</i>	input matrix to calculate low rank approximation
<i>r</i>	desired rank, must be less than the minimum between the number of columns and rows of A
<i>B</i>	output left bilateral matrix
<i>C</i>	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

<i>A</i>	Matrix to calculate pseudoinverse
<i>Ap</i>	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

<i>t_data_x</i>	the original source images. The training data is a vector of vectors containing a vectorized form of an image
<i>t_data_c</i>	the images with a noise. The training data is a vector of vectors containing a vectorized form of an image
<i>rank</i>	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

<code>t_data↔ _x</code>	the original source images. The training data is an <code>arma::mat</code> where each column is a vectorized form of a training image
<code>t_data↔ _c</code>	the images with a noise applied. The training data is an <code>arma::mat</code> where each column is a vectorized form of a training image
<code>rank</code>	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

<a href="#">FD_IMAGE</a>	an image file dialog
<a href="#">FD_ZIP</a>	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

<i>mimeTypeFilters</i>	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

<i>list</i>	list of items to filter
<i>filter_string</i>	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

<i>rows</i>	amount of rows
<i>cols</i>	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**

<i>list</i>	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**

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



## Chapter 6

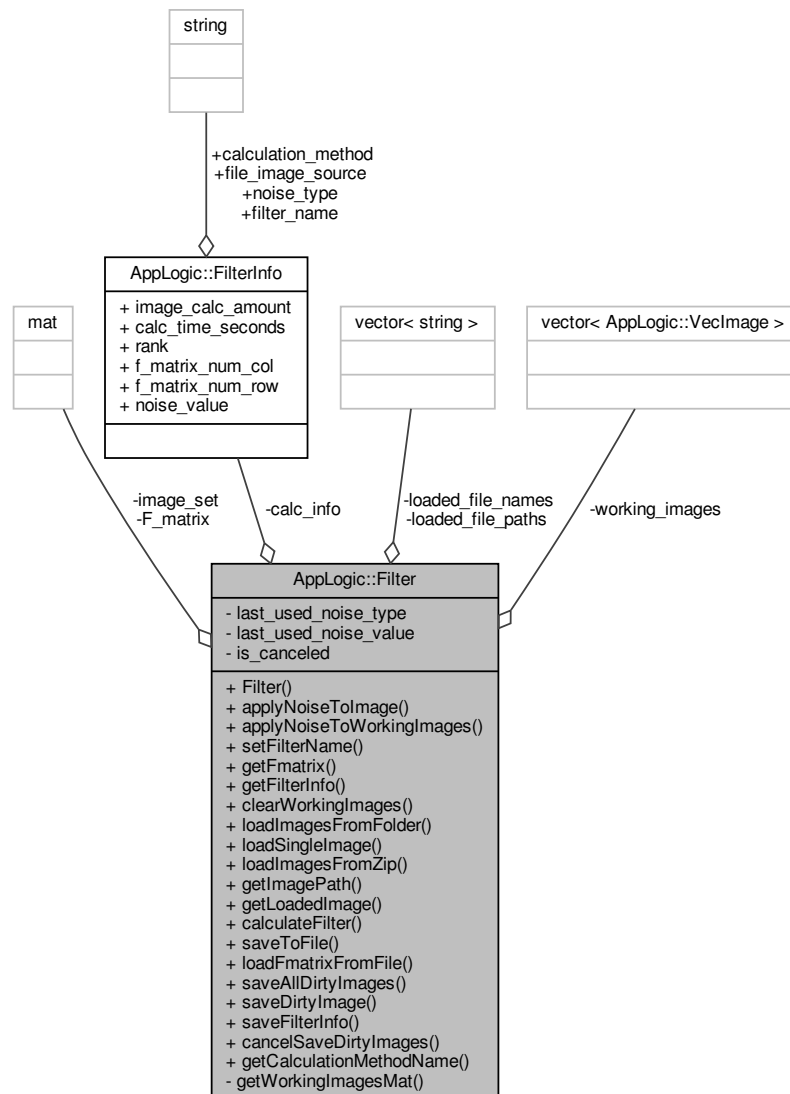
# Class Documentation

### 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

<i>image_id</i>	the index of the loaded image to apply noise to
<i>noise_value</i>	percentage value (0-99) of amount of noise to apply
<i>noise_type</i>	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**

<i>rank</i>	the rank to use for filter calculation. Must be a value between 1 and the minimum between the amount of images used and the size of the vectorized image.
<i>calc_method</i>	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

<i>calc_method</i>	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**

<i>index</i>	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**

<i>index</i>	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**

<i>file_path</i>	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**

<i>folder_path</i>	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**

<i>file_path</i>	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

<i>file_path</i>	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

<i>folder_path</i>	path where the images will be saved
--------------------	-------------------------------------

##### Returns

true if images are saved correctly  
false if an error 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**

<i>image_id</i>	index of the image to apply noise to and save
<i>folder_path</i>	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**

<i>folder_path</i>	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**

<i>file_path</i>	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

<i>name</i>	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](#)

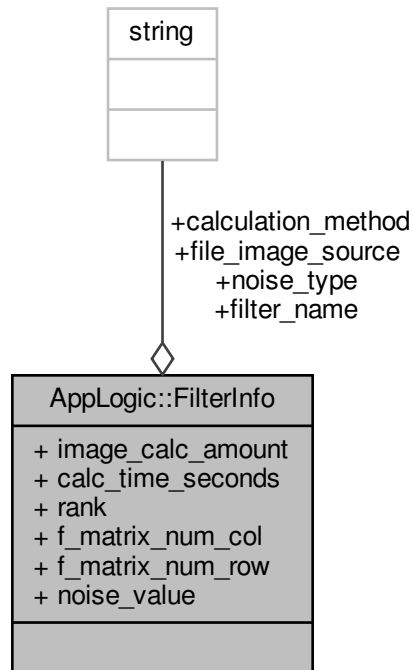


## 6.2 AppLogic::FilterInfo Struct Reference

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 or 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 or 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:

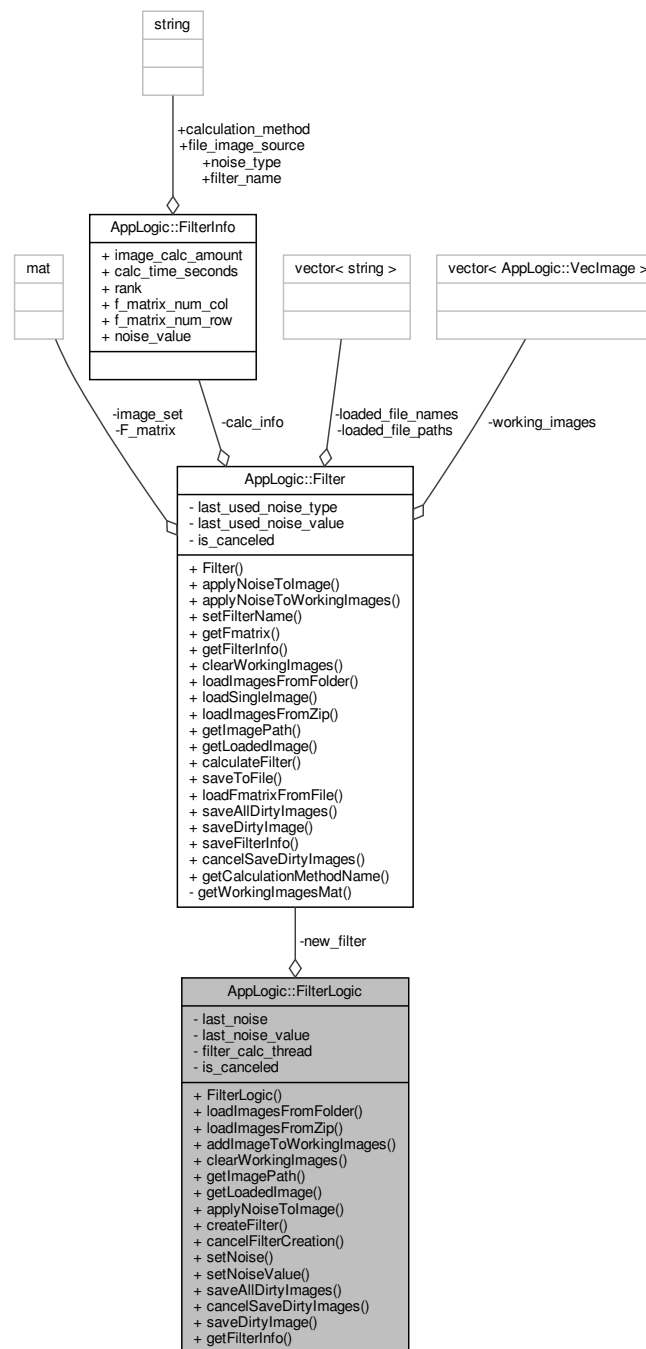
- [FilterInfo.hpp](#)

## 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

<i>file_path</i>	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

<i>image_id</i>	the index of the loaded image to apply noise to
<i>noise</i>	type of noise to apply to the image
<i>noise_value</i>	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**

<i>name</i>	name of the filter. A file is created with the filter name, no validation is done on the name
<i>rank</i>	the rank to use for filter calculation. Must be a value between 1 and the minimum between the amount of images used and the size of the vectorized image.
<i>calc_method</i>	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 for current filter.

**Returns**

FilterInfo\* Object containing the calculation information for the current 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**

<i>index</i>	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**

<i>index</i>	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**

<i>folder_path</i>	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**

<i>file_path</i>	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

<i>folder_path</i>	path where the images will be saved
--------------------	-------------------------------------

##### Returns

true if images are saved correctly  
false if an error 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

<i>image_id</i>	index of the image to apply noise to and save
<i>folder_path</i>	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

<i>n_type</i>	type of noise
---------------	---------------

## 6.3.3.15 setNoiseValue()

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

sets the last noise value used

## Parameters

<i>val</i>	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:

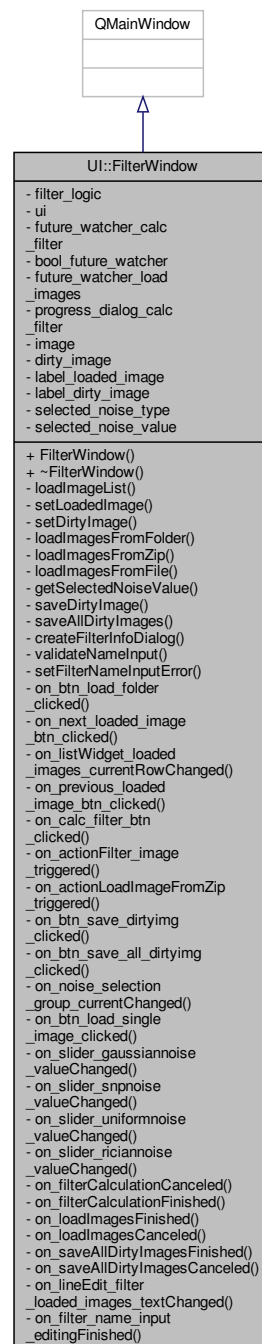
- [FilterLogic.hpp](#)
- [FilterLogic.cpp](#)

## 6.4 UI::FilterWindow Class Reference

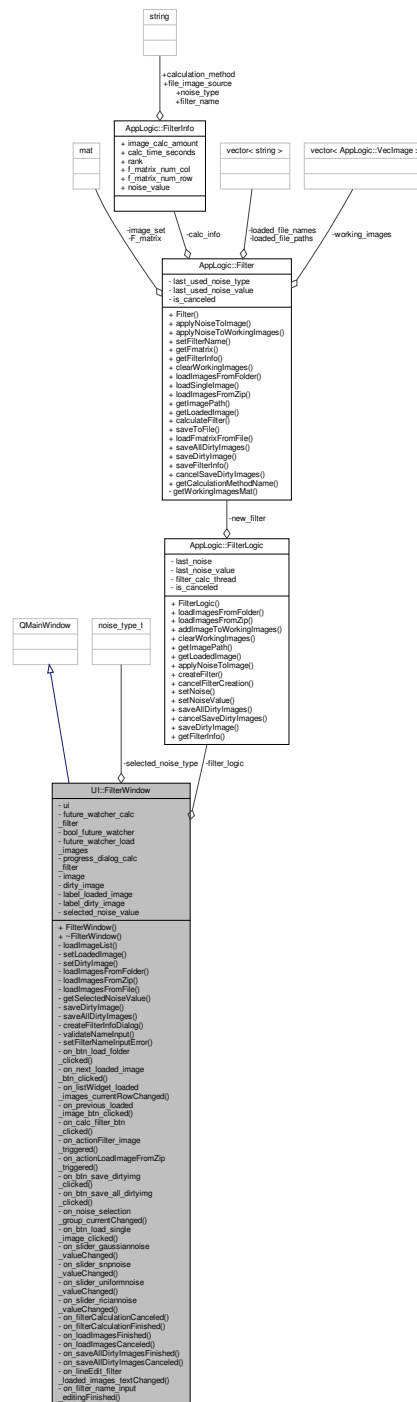
Window to create filters.

```
#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

<i>parent</i>	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

<i>noise_type</i>	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

<i>loaded_filenames</i>	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**

<i>filenames</i>	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**

<i>folder_path</i>	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

<i>folder_path</i>	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

<i>fileName</i>	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

<i>fileName</i>	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**

<i>error_message</i>	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**

<i>index</i>	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 illegal.

##### 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:

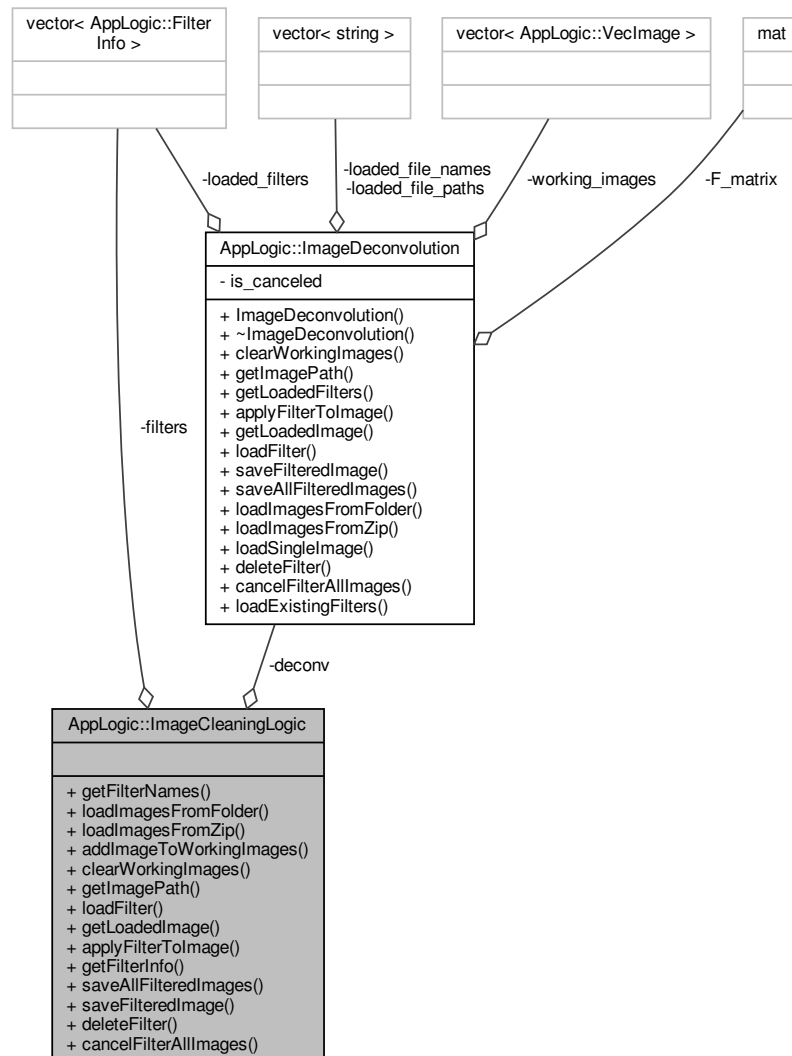
- [FilterWindow.h](#)
- [FilterWindow.cpp](#)

## 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

<code>file_path</code>	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.

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

**Parameters**

<i>image_id</i>	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 \*.info and \*.mat files that ath the name of the filter specified by `index` in filters.

**Parameters**

<i>index</i>	index of the <a href="#">FilterInfo</a> 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**

<i>index</i>	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**

<i>index</i>	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**

<i>filter_id</i>	index of the <a href="#">FilterInfo</a> in <code>filters</code> 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

<i>folder_path</i>	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

<i>file_path</i>	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 loaded filter and saves them to the folder path provided as PNG (.png) images. This operation does not modify the loaded images.

#### Parameters

<i>folder_path</i>	path where the images will be saved
--------------------	-------------------------------------



**Returns**

true if images are saved correctly  
false if an error 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**

<i>image_id</i>	index of the image to apply noise to and save
<i>folder_path</i>	name of the file to save the dirty image

**Returns**

true if image is saved correctly  
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:

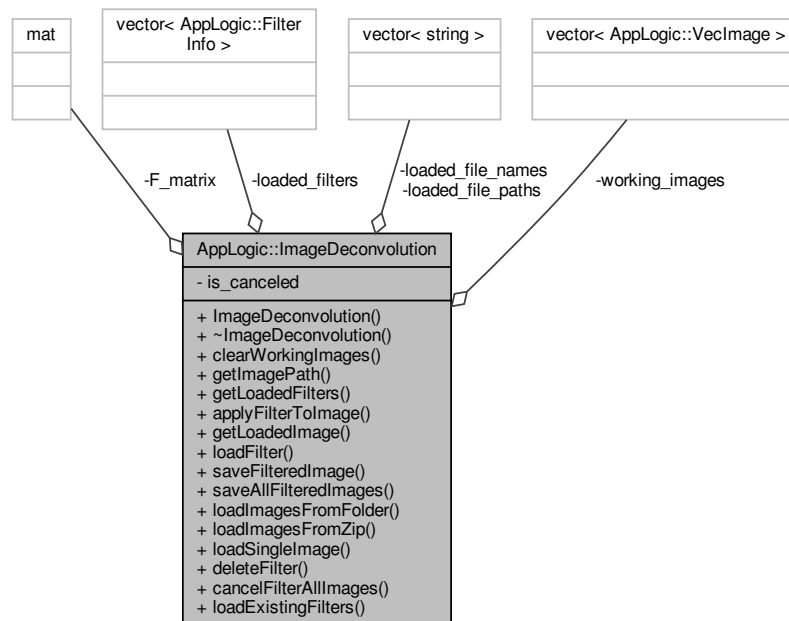
- [ImageCleaningLogic.hpp](#)
- [ImageCleaningLogic.cpp](#)

## 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.

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

#### Parameters

<i>image_id</i>	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

<i>index</i>	index of the <a href="#">FilterInfo</a> 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

<i>index</i>	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**

<i>index</i>	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**

<i>id</i>	index of the <a href="#">FilterInfo</a> in <code>loaded_filters</code> 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**

<i>folder_path</i>	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**

<i>file_path</i>	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**

<i>file_path</i>	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 loaded filter and saves them to the folder path provided as PNG (.png) images. This operation does not modify the loaded images.

#### Parameters

<i>folder_path</i>	path where the images will be saved
--------------------	-------------------------------------

#### Returns

true if images are saved correctly  
false if an error 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

<i>image_id</i>	index of the image to apply noise to and save
<i>folder_path</i>	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:

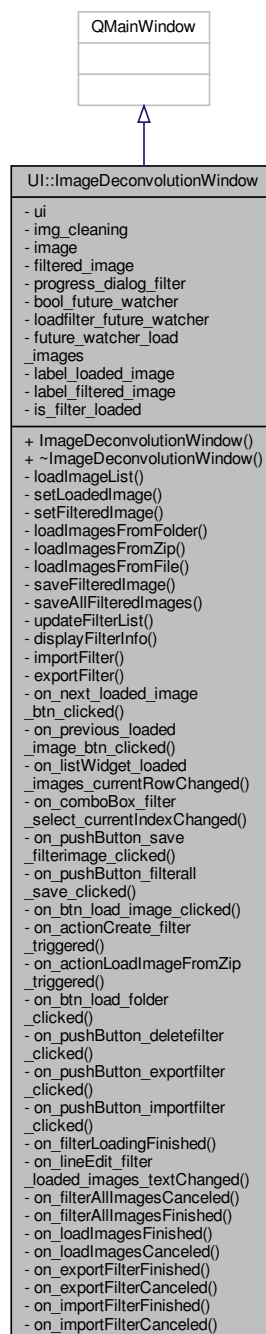
- [ImageDeconvolution.hpp](#)
- [ImageDeconvolution.cpp](#)

## 6.7 UI::ImageDeconvolutionWindow Class Reference

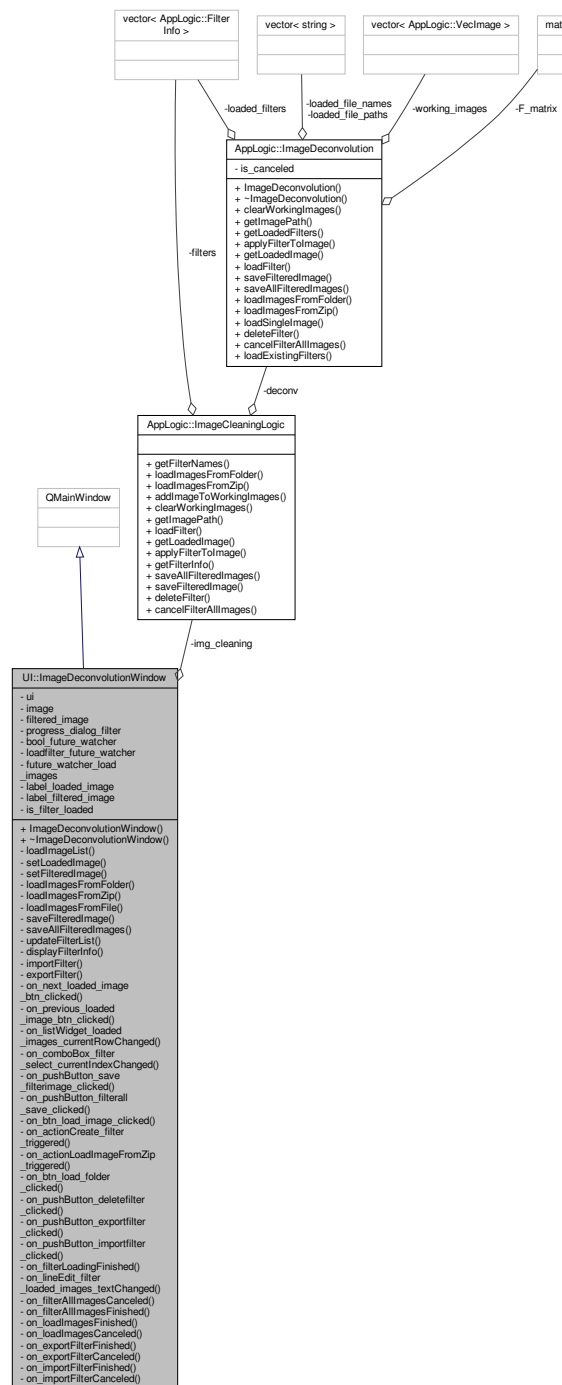
Window to filter images.

```
#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

<i>parent</i>	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

<i>fi</i>	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

<i>export_filename</i>	
------------------------	--

### 6.7.3.3 importFilter()

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

imports filter from a zip export file

#### Parameters

<i>import_filename</i>	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

<i>loaded_filenames</i>	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

<i>filenames</i>	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

<i>folder_path</i>	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

<i>folder_path</i>	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

<i>fileName</i>	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

<i>fileName</i>	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

<i>index</i>	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

<i>index</i>	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:

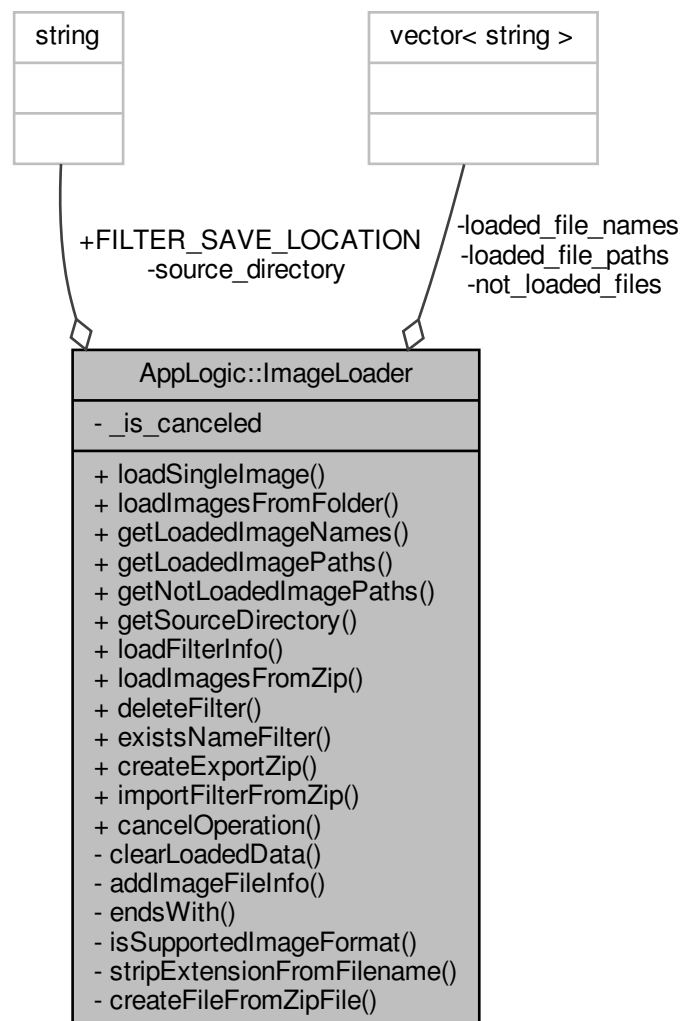
- [ImageDeconvolutionWindow.h](#)
- [ImageDeconvolutionWindow.cpp](#)

## 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 suffix*
- 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

<i>full_path</i>	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

<i>status</i>	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

<i>export_filename</i>	name of the zip file to create
<i>filter_name</i>	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

<i>filename</i>	
<i>stream</i>	
<i>zip_file</i>	

##### 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

<i>filter_name</i>	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 suffix

**Parameters**

<i>str</i>	string to check
<i>suffix</i>	suffix to check

**Returns**

true if string ends in suffix  
false if 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**

<i>filter_name</i>	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

<i>import_filename</i>	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

<i>str</i>	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

<i>folder_path</i>	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

<i>file_path</i>	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

<i>filename</i>	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

<i>filename</i>	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:

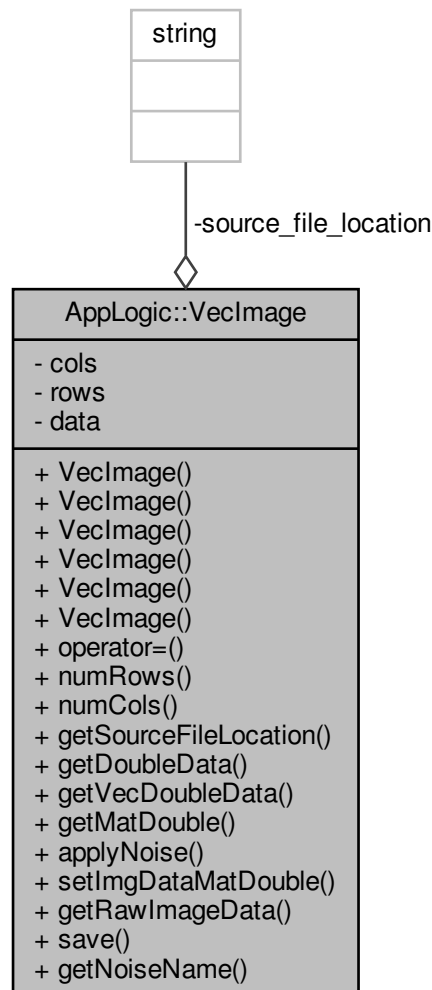
- [ImageLoader.hpp](#)
- [ImageLoader.cpp](#)

## 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

<i>filename</i>	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

<i>img_data</i>	vector containig normalized values [0,1] of the pixel data of the image
<i>width</i>	horizontal size of the image
<i>height</i>	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**

<i>img_data</i>	vector containig normalized values [0,1] of the pixel data of the image
<i>width</i>	horizontal size of the image
<i>height</i>	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**

<i>cImg</i>	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**

<i>vImg</i>	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

<i>noise_value</i>	value representing the percentage of the noise to be applied [1,99]
<i>noise</i>	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**

<i>noise_type</i>	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

<i>image</i>	
--------------	--

##### Returns

VecImage&

#### 6.9.3.11 save()

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

saves an image to file as PNG

**Parameters**

<i>filename</i>	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**

<i>new_data</i>	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](#)



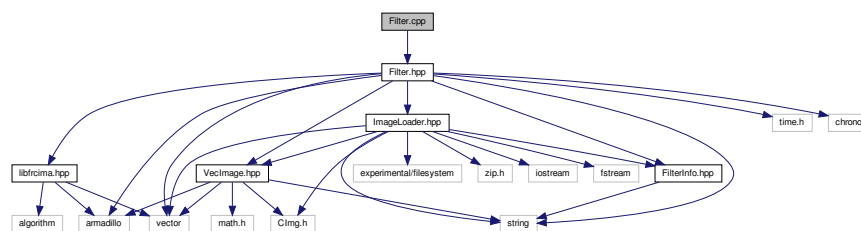
## Chapter 7

# File Documentation

### 7.1 Filter.cpp File Reference

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

Include dependency graph for Filter.cpp:



### Namespaces

- [AppLogic](#)

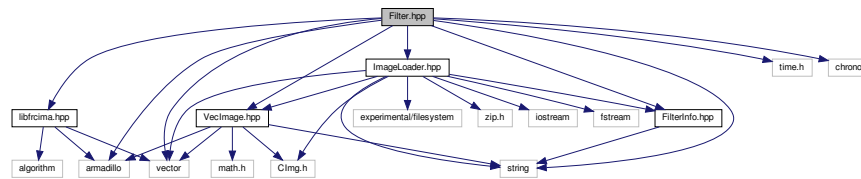
### 7.2 Filter.hpp File Reference

Contains `Filter` class in charge of creating filters.

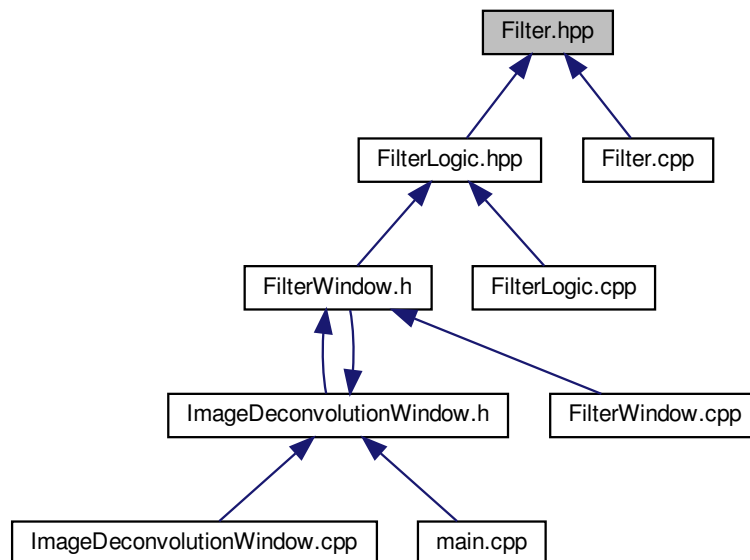
```
#include <vector>
#include <string>
#include <armadillo>
#include <time.h>
#include <chrono>
#include "FilterInfo.hpp"
#include "VecImage.hpp"
#include "ImageLoader.hpp"
```

```
#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"
```

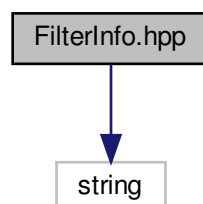
suffix used in saved noisy image filenames

## 7.3 FilterInfo.hpp File Reference

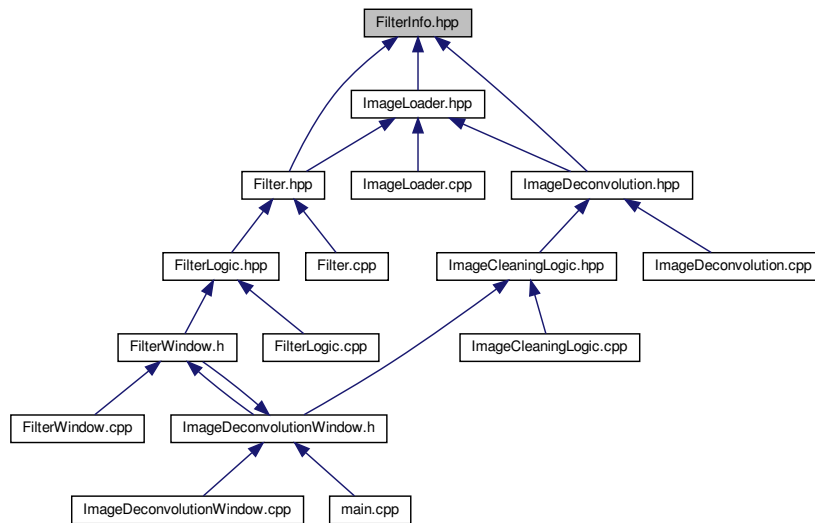
Contains struct to store filter calculation information and the members to save this information into files.

```
#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

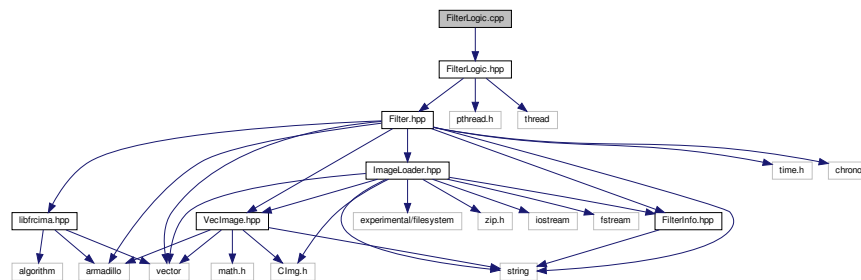
#### Date

2021-06-13



## 7.4 FilterLogic.cpp File Reference

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



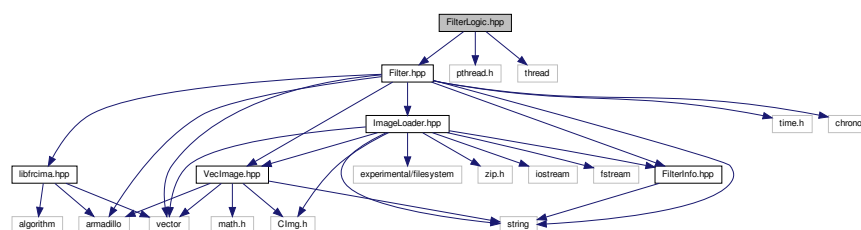
### Namespaces

- [AppLogic](#)

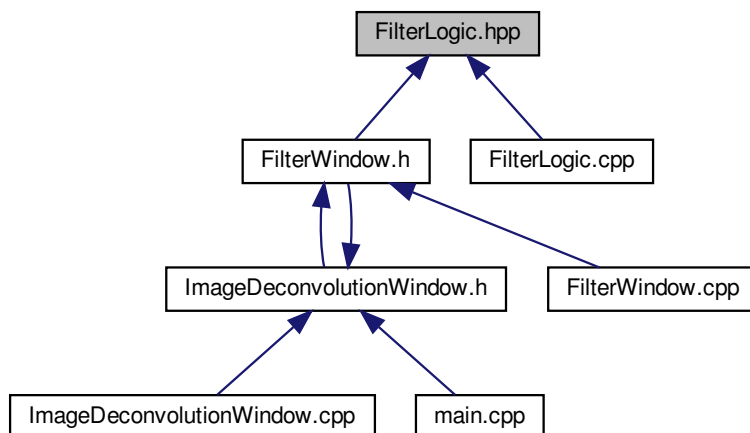
## 7.5 FilterLogic.hpp File Reference

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

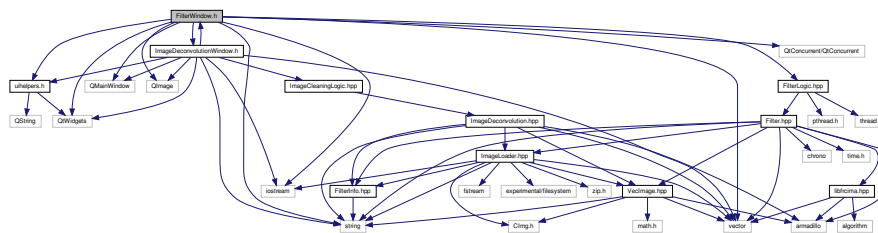


## 7.7 FilterWindow.h File Reference

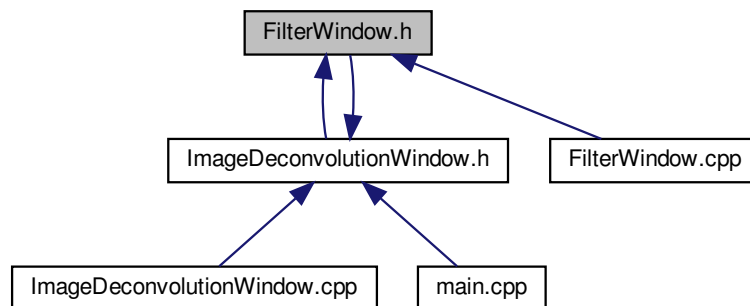
Contains the FilterWindow class which holds the [UI](#) to create filters.

```
#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

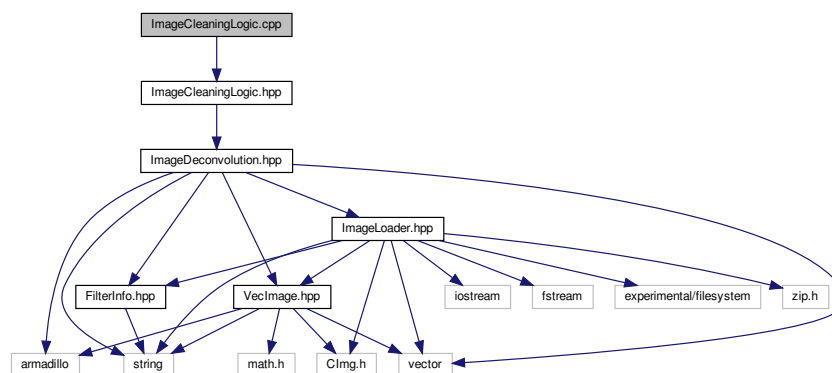
**Date**

2021-06-13

## 7.8 ImageCleaningLogic.cpp File Reference

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

Include dependency graph for ImageCleaningLogic.cpp:



### Namespaces

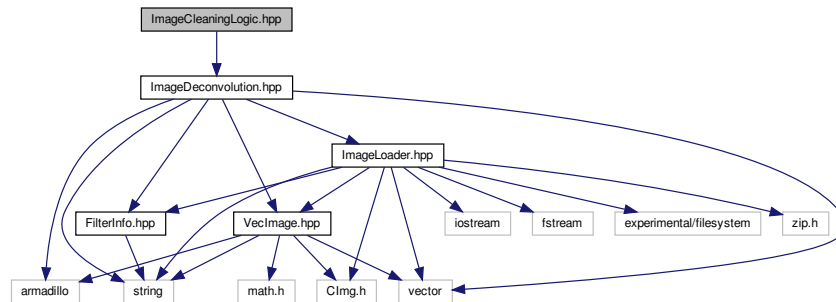
- [AppLogic](#)

## 7.9 ImageCleaningLogic.hpp File Reference

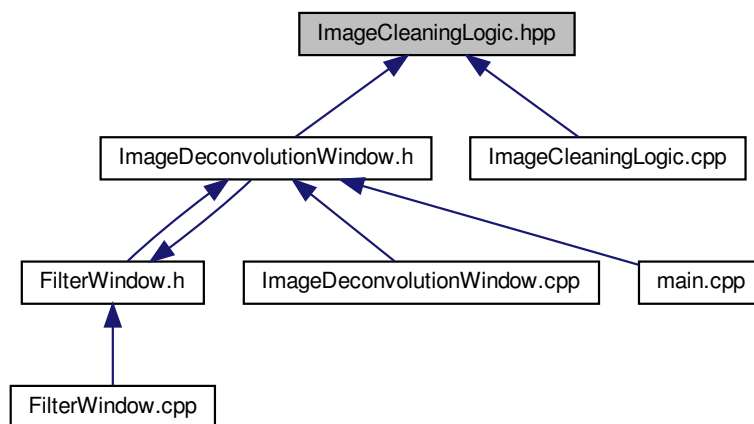
Contains the API to be able to apply filters to images.

```
#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

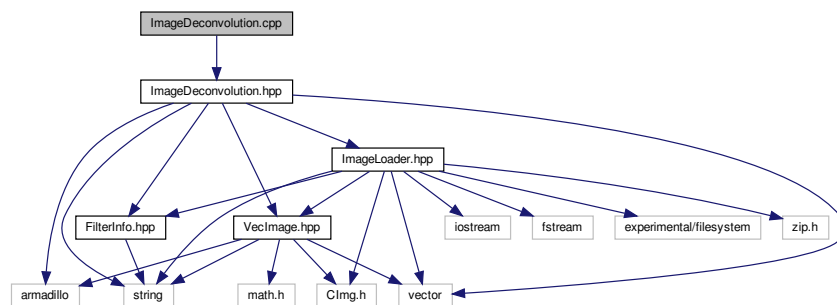
**Date**

2021-06-13

## 7.10 ImageDeconvolution.cpp File Reference

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

Include dependency graph for ImageDeconvolution.cpp:

**Namespaces**

- [AppLogic](#)

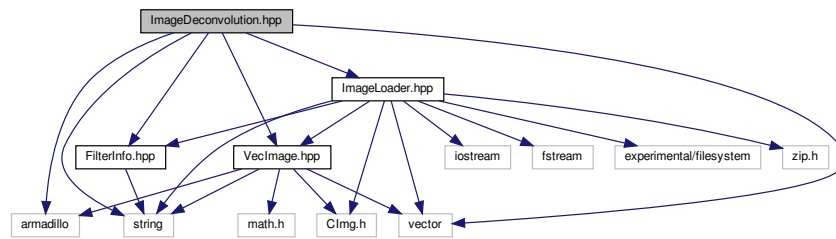
## 7.11 ImageDeconvolution.hpp File Reference

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

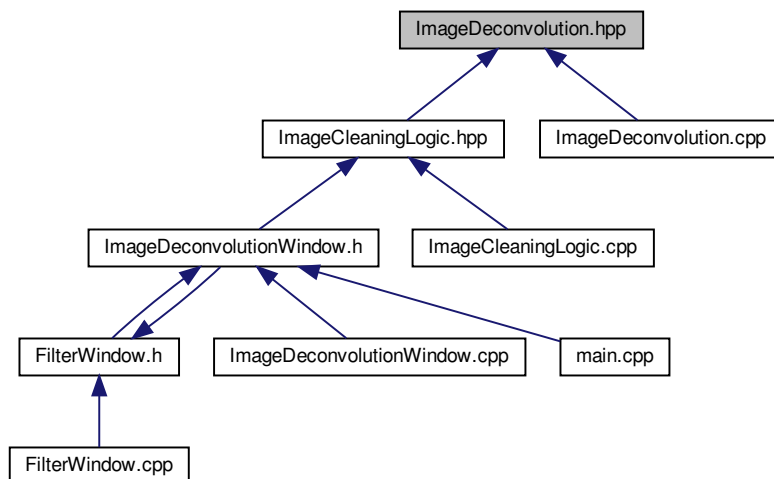
```
#include <vector>
#include <string>
#include <armadillo>
#include "FilterInfo.hpp"
#include "VecImage.hpp"
```

```
#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*





## Macros

- `#define ELIDE_WIDTH 400`

### 7.12.1 Macro Definition Documentation

#### 7.12.1.1 ELIDE\_WIDTH

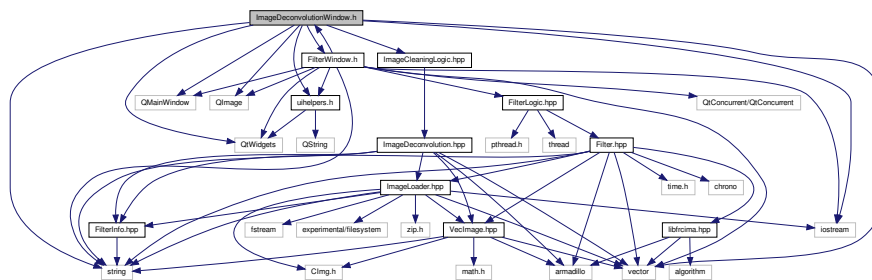
```
#define ELIDE_WIDTH 400
```

## 7.13 ImageDeconvolutionWindow.h File Reference

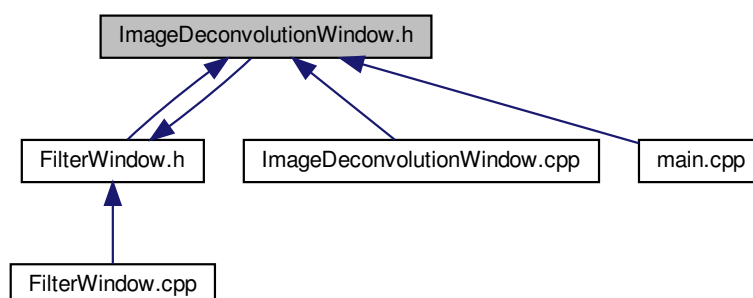
Contains `ImageDeconvolutionWindow` class which holds the [UI](#) to apply filters to images.

```
#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

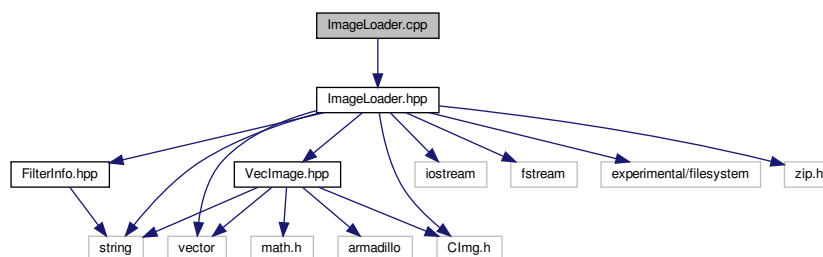
#### Copyright

Copyright (c) 2021

## 7.14 ImageLoader.cpp File Reference

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

Include dependency graph for `ImageLoader.cpp`:



## Namespaces

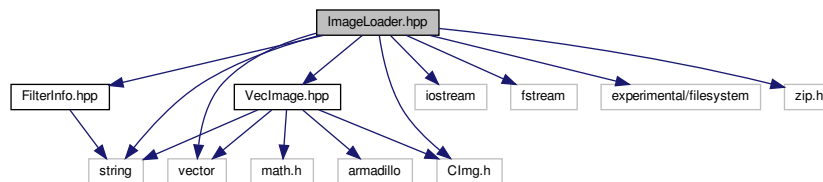
- [AppLogic](#)

## 7.15 ImageLoader.hpp File Reference

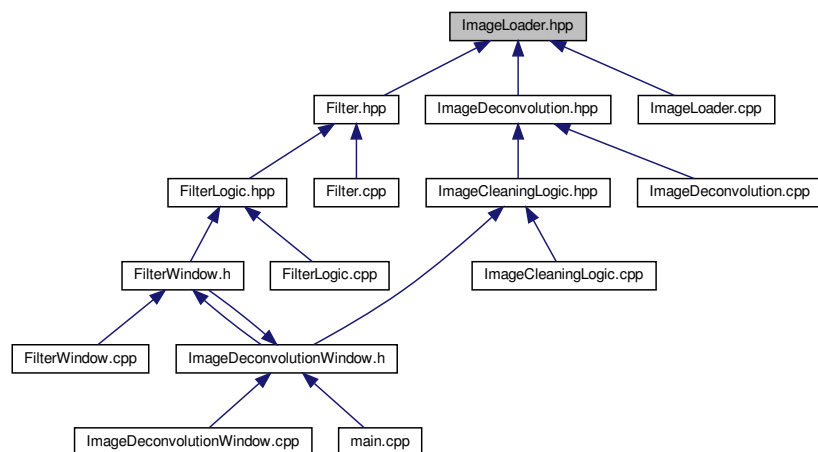
Contains ImageLoader class in charge of loading images and filter information.

```
#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

Copyright (c) 2021

### 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

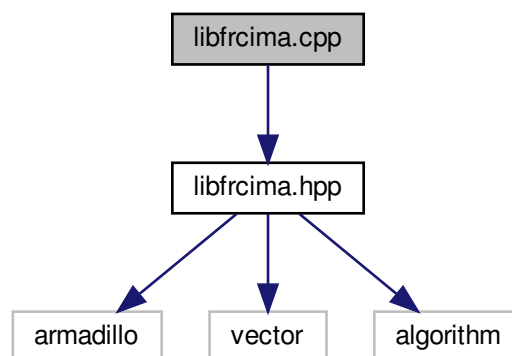
```
#define ZIP_FILE_BUFF_SIZE 4096
```

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

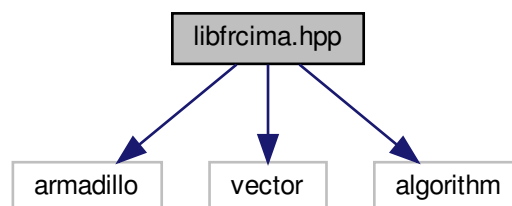
- void `frcima::calculate_error` (mat &A, const mat &C, const mat &X)  
*calculates the error from the calculated filter*
- bool `frcima::low_rank_approx` (const mat &A, const size\_t r, mat &B, mat &C)  
*calculates low rank matrix approximation using the modified bilateral random projections (MBRP) 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

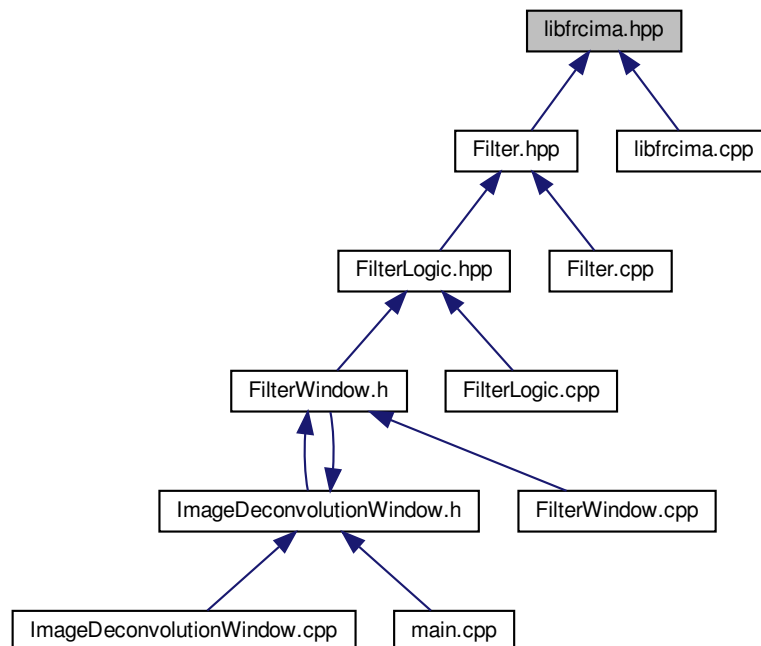
Contains the RCIMA and fast-RCIMA methods.

```
#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

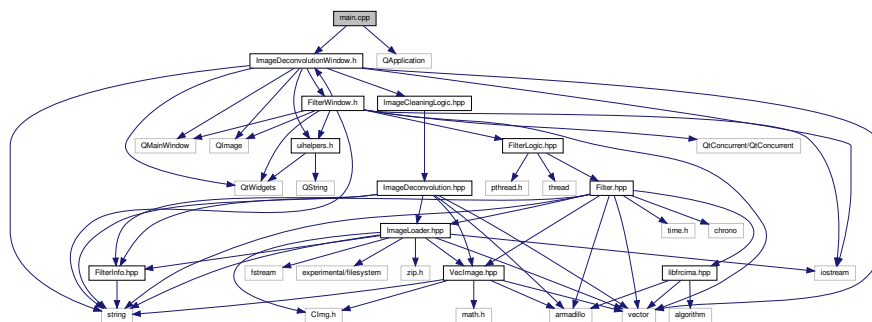
## 7.18 main.cpp File Reference

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()`

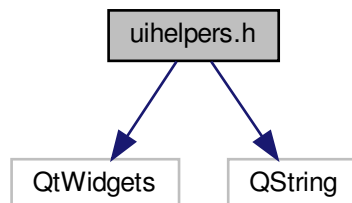
```
int main (  
    int argc,  
    char * argv[] )
```

## 7.19 uihelpers.h File Reference

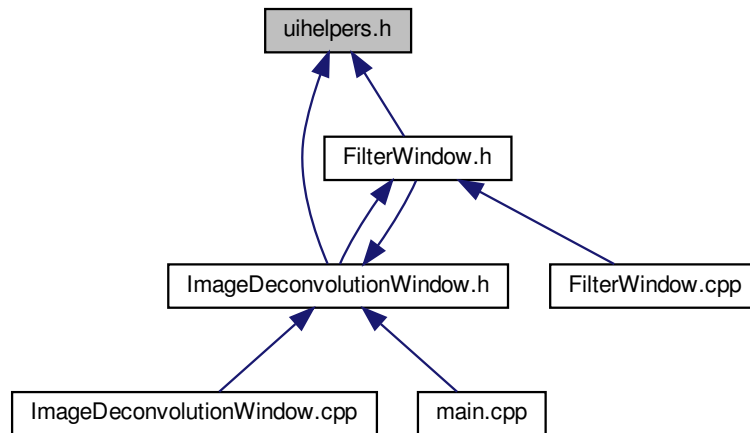
Contains helper functions for the Window classes.

```
#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=QFileDialog::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

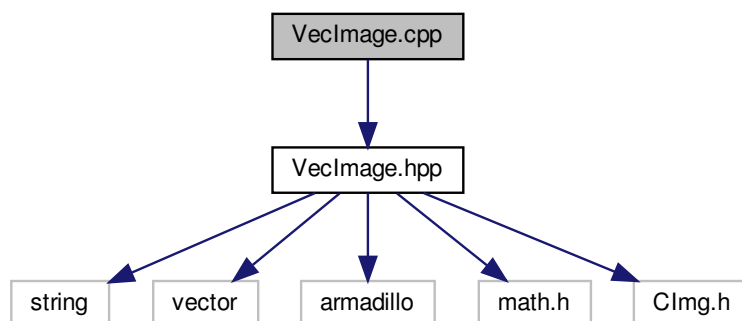
```
#define STANDARD_MESSAGE_TIME 2000
```

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

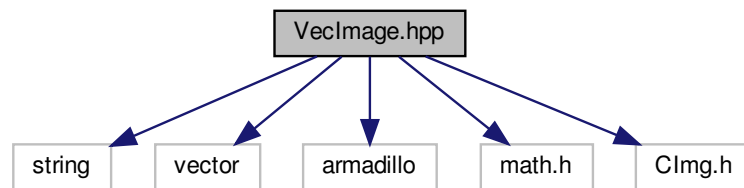
```
#define VECIMG_NORM 255
```

## 7.21 VecImage.hpp File Reference

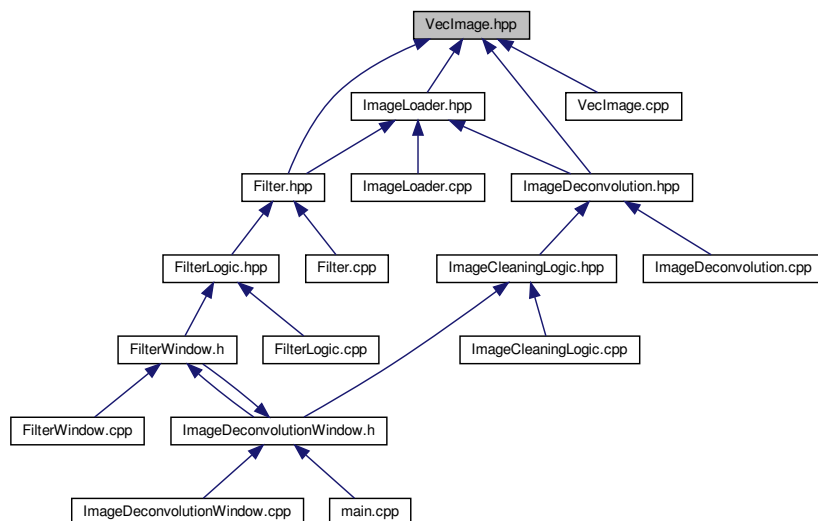
Contains the `VecImage` class used to manipulate images.

```
#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

- `_calculated_error`
    - `frcima`, [16](#)
  - `_is_canceled`
    - `AppLogic::ImageLoader`, [97](#)
  - `~FilterWindow`
    - `UI::FilterWindow`, [48](#)
  - `~ImageDeconvolution`
    - `AppLogic::ImageDeconvolution`, [67](#)
  - `~ImageDeconvolutionWindow`
    - `UI::ImageDeconvolutionWindow`, [78](#)
- `addImageFileInfo`
  - `AppLogic::ImageLoader`, [90](#)
- `addImageToWorkingImages`
  - `AppLogic::FilterLogic`, [38](#)
  - `AppLogic::ImageCleaningLogic`, [60](#)
- `allSupportedFormatsString`
  - `uihelp`, [18](#)
- `AppLogic`, [9](#)
  - `calc_method_t`, [10](#)
  - `noise_type_t`, [10](#)
  - `operator<<`, [10](#)
  - `operator>>`, [11](#)
- `AppLogic::Filter`, [21](#)
  - `applyNoiseToImage`, [24](#)
  - `applyNoiseToWorkingImages`, [25](#)
  - `calc_info`, [31](#)
  - `calculateFilter`, [25](#)
  - `cancelSaveDirtyImages`, [25](#)
  - `clearWorkingImages`, [25](#)
  - `F_matrix`, [31](#)
  - `Filter`, [24](#)
  - `getCalculationMethodName`, [25](#)
  - `getFilterInfo`, [26](#)
  - `getFmatrix`, [26](#)
  - `getImagePath`, [26](#)
  - `getLoadedImage`, [27](#)
  - `getWorkingImagesMat`, [27](#)
  - `image_set`, [31](#)
  - `is_canceled`, [31](#)
  - `last_used_noise_type`, [31](#)
  - `last_used_noise_value`, [32](#)
  - `loadFmatrixFromFile`, [27](#)
  - `loadImagesFromFolder`, [28](#)
  - `loadImagesFromZip`, [28](#)
  - `loadSingleImage`, [28](#)
  - `loaded_file_names`, [32](#)
  - `loaded_file_paths`, [32](#)
  - `saveAllDirtyImages`, [29](#)
  - `saveDirtyImage`, [29](#)
  - `saveFilterInfo`, [30](#)
  - `saveToFile`, [30](#)
  - `setFilterName`, [30](#)
  - `working_images`, [32](#)
- `AppLogic::FilterInfo`, [33](#)
  - `calc_time_seconds`, [34](#)
  - `calculation_method`, [34](#)
  - `f_matrix_num_col`, [34](#)
  - `f_matrix_num_row`, [34](#)
  - `file_image_source`, [34](#)
  - `filter_name`, [35](#)
  - `image_calc_amount`, [35](#)
  - `noise_type`, [35](#)
  - `noise_value`, [35](#)
  - `rank`, [35](#)
- `AppLogic::FilterLogic`, [36](#)
  - `addImageToWorkingImages`, [38](#)
  - `applyNoiseToImage`, [38](#)
  - `cancelFilterCreation`, [39](#)
  - `cancelSaveDirtyImages`, [39](#)
  - `clearWorkingImages`, [39](#)
  - `createFilter`, [39](#)
  - `filter_calc_thread`, [43](#)
  - `FilterLogic`, [38](#)
  - `getFilterInfo`, [40](#)
  - `getImagePath`, [40](#)
  - `getLoadedImage`, [40](#)
  - `is_canceled`, [43](#)
  - `last_noise`, [43](#)
  - `last_noise_value`, [43](#)
  - `loadImagesFromFolder`, [41](#)
  - `loadImagesFromZip`, [41](#)
  - `new_filter`, [44](#)
  - `saveAllDirtyImages`, [41](#)
  - `saveDirtyImage`, [42](#)
  - `setNoise`, [42](#)
  - `setNoiseValue`, [43](#)
- `AppLogic::ImageCleaningLogic`, [58](#)
  - `addImageToWorkingImages`, [60](#)
  - `applyFilterToImage`, [61](#)
  - `cancelFilterAllImages`, [61](#)
  - `clearWorkingImages`, [61](#)
  - `deconv`, [65](#)
  - `deleteFilter`, [61](#)
  - `filters`, [65](#)
  - `getFilterInfo`, [62](#)
  - `getFilterNames`, [62](#)
  - `getImagePath`, [62](#)
  - `getLoadedImage`, [63](#)

- loadFilter, 63
- loadImagesFromFolder, 63
- loadImagesFromZip, 64
- saveAllFilteredImages, 64
- saveFilteredImage, 65
- AppLogic::ImageDeconvolution, 66
  - ~ImageDeconvolution, 67
  - applyFilterToImage, 68
  - cancelFilterAllImages, 68
  - clearWorkingImages, 68
  - deleteFilter, 68
  - F\_matrix, 73
  - getImagePath, 69
  - getLoadedFilters, 69
  - getLoadedImage, 69
  - ImageDeconvolution, 67
  - is\_canceled, 73
  - loadExistingFilters, 70
  - loadFilter, 70
  - loadImagesFromFolder, 70
  - loadImagesFromZip, 72
  - loadSingleImage, 72
  - loaded\_file\_names, 74
  - loaded\_file\_paths, 74
  - loaded\_filters, 74
  - saveAllFilteredImages, 72
  - saveFilteredImage, 73
  - working\_images, 74
- AppLogic::ImageLoader, 88
  - \_is\_canceled, 97
  - addImageFileInfo, 90
  - cancelOperation, 91
  - clearLoadedData, 91
  - createExportZip, 91
  - createFileFromZipFile, 91
  - deleteFilter, 92
  - endsWith, 92
  - existsNameFilter, 93
  - FILTER\_SAVE\_LOCATION, 98
  - getLoadedImageNames, 93
  - getLoadedImagePaths, 93
  - getNotLoadedImagePaths, 93
  - getSourceDirectory, 94
  - importFilterFromZip, 94
  - isSupportedImageFormat, 94
  - loadFilterInfo, 95
  - loadImagesFromFolder, 95
  - loadImagesFromZip, 95
  - loadSingleImage, 97
  - loaded\_file\_names, 98
  - loaded\_file\_paths, 98
  - not\_loaded\_files, 98
  - source\_directory, 98
  - stripExtensionFromFilename, 97
- AppLogic::VecImage, 99
  - applyNoise, 102
  - cols, 106
  - data, 106
  - getDoubleData, 103
  - getMatDouble, 103
  - getNoiseName, 103
  - getRawImageData, 104
  - getSourceFileLocation, 104
  - getVecDoubleData, 104
  - numCols, 104
  - numRows, 105
  - operator=, 105
  - rows, 106
  - save, 105
  - setImgDataMatDouble, 106
  - source\_file\_location, 106
  - VecImage, 101, 102
- applyFilterToImage
  - AppLogic::ImageCleaningLogic, 61
  - AppLogic::ImageDeconvolution, 68
- applyNoise
  - AppLogic::VecImage, 102
- applyNoiseToImage
  - AppLogic::Filter, 24
  - AppLogic::FilterLogic, 38
- applyNoiseToWorkingImages
  - AppLogic::Filter, 25
- bool\_future\_watcher
  - UI::FilterWindow, 56
  - UI::ImageDeconvolutionWindow, 86
- calc\_info
  - AppLogic::Filter, 31
- calc\_method\_t
  - AppLogic, 10
- calc\_time\_seconds
  - AppLogic::FilterInfo, 34
- calculate\_error
  - frcima, 12
- calculateFilter
  - AppLogic::Filter, 25
- calculation\_method
  - AppLogic::FilterInfo, 34
- cancelFilterAllImages
  - AppLogic::ImageCleaningLogic, 61
  - AppLogic::ImageDeconvolution, 68
- cancelFilterCreation
  - AppLogic::FilterLogic, 39
- cancelOperation
  - AppLogic::ImageLoader, 91
- cancelSaveDirtyImages
  - AppLogic::Filter, 25
  - AppLogic::FilterLogic, 39
- cimg\_display
  - VecImage.hpp, 135
- cimg\_use\_jpeg
  - VecImage.hpp, 135
- cimg\_use\_png
  - VecImage.hpp, 135
- clearLoadedData
  - AppLogic::ImageLoader, 91

- clearWorkingImages
  - AppLogic::Filter, 25
  - AppLogic::FilterLogic, 39
  - AppLogic::ImageCleaningLogic, 61
  - AppLogic::ImageDeconvolution, 68
- cols
  - AppLogic::VecImage, 106
- createExportZip
  - AppLogic::ImageLoader, 91
- createFileFromZipFile
  - AppLogic::ImageLoader, 91
- createFilter
  - AppLogic::FilterLogic, 39
- createFilterInfoDialog
  - UI::FilterWindow, 49
- DIRTY\_IMAGE\_SUFFIX
  - Filter.hpp, 111
- data
  - AppLogic::VecImage, 106
- deconv
  - AppLogic::ImageCleaningLogic, 65
- deleteFilter
  - AppLogic::ImageCleaningLogic, 61
  - AppLogic::ImageDeconvolution, 68
  - AppLogic::ImageLoader, 92
- dirty\_image
  - UI::FilterWindow, 56
- displayFilterInfo
  - UI::ImageDeconvolutionWindow, 79
- ELEMENT\_NUMBER\_MILLION\_CUTOFF
  - uihelpers.h, 132
- ELEMENT\_NUMBER\_THOUSAND\_CUTOFF
  - uihelpers.h, 132
- ELIDE\_WIDTH
  - ImageDeconvolutionWindow.cpp, 122
- endsWith
  - AppLogic::ImageLoader, 92
- existsNameFilter
  - AppLogic::ImageLoader, 93
- exportFilter
  - UI::ImageDeconvolutionWindow, 79
- F\_matrix
  - AppLogic::Filter, 31
  - AppLogic::ImageDeconvolution, 73
- f\_matrix\_num\_col
  - AppLogic::FilterInfo, 34
- f\_matrix\_num\_row
  - AppLogic::FilterInfo, 34
- FILT\_IMAGE\_SUFFIX
  - ImageDeconvolution.hpp, 121
- FILTER\_FILE\_EXTENSION
  - ImageLoader.hpp, 125
- FILTER\_INFO\_EXTENSION
  - ImageLoader.hpp, 125
- FILTER\_SAVE\_LOCATION
  - AppLogic::ImageLoader, 98
- fast\_rcima
  - frcima, 12
- file\_dialog\_type\_t
  - uihelp, 17
- file\_image\_source
  - AppLogic::FilterInfo, 34
- Filter
  - AppLogic::Filter, 24
- Filter.cpp, 109
- Filter.hpp, 109
  - DIRTY\_IMAGE\_SUFFIX, 111
- filter\_calc\_thread
  - AppLogic::FilterLogic, 43
- filter\_logic
  - UI::FilterWindow, 56
- filter\_name
  - AppLogic::FilterInfo, 35
- FilterInfo.hpp, 111
- filterListItems
  - uihelp, 18
- FilterLogic
  - AppLogic::FilterLogic, 38
- FilterLogic.cpp, 113
- FilterLogic.hpp, 113
- FilterWindow
  - UI::FilterWindow, 48
- FilterWindow.cpp, 115
  - SLASHES, 115
  - WINDOWS\_DEVICES, 115
- FilterWindow.h, 116
- filtered\_image
  - UI::ImageDeconvolutionWindow, 86
- filters
  - AppLogic::ImageCleaningLogic, 65
- frcima, 11
  - \_calculated\_error, 16
  - calculate\_error, 12
  - fast\_rcima, 12
  - generate\_training\_matrices, 13
  - low\_rank, 13
  - low\_rank\_approx, 14
  - pseudo\_inverse\_tpm, 14
  - rcima, 15
- future\_watcher\_calc\_filter
  - UI::FilterWindow, 56
- future\_watcher\_load\_images
  - UI::FilterWindow, 56
  - UI::ImageDeconvolutionWindow, 86
- generate\_training\_matrices
  - frcima, 13
- generateSizeMessage
  - uihelp, 18
- getCalculationMethodName
  - AppLogic::Filter, 25
- getDoubleData
  - AppLogic::VecImage, 103
- getFilterInfo
  - AppLogic::Filter, 26

- AppLogic::FilterLogic, 40
- AppLogic::ImageCleaningLogic, 62
- getFilterNames
  - AppLogic::ImageCleaningLogic, 62
- getFmatrix
  - AppLogic::Filter, 26
- getImagePath
  - AppLogic::Filter, 26
  - AppLogic::FilterLogic, 40
  - AppLogic::ImageCleaningLogic, 62
  - AppLogic::ImageDeconvolution, 69
- getLoadedFilters
  - AppLogic::ImageDeconvolution, 69
- getLoadedImage
  - AppLogic::Filter, 27
  - AppLogic::FilterLogic, 40
  - AppLogic::ImageCleaningLogic, 63
  - AppLogic::ImageDeconvolution, 69
- getLoadedImageNames
  - AppLogic::ImageLoader, 93
- getLoadedImagePaths
  - AppLogic::ImageLoader, 93
- getMatDouble
  - AppLogic::VecImage, 103
- getNoiseName
  - AppLogic::VecImage, 103
- getNotLoadedImagePaths
  - AppLogic::ImageLoader, 93
- getRawImageData
  - AppLogic::VecImage, 104
- getSelectedNoiseValue
  - UI::FilterWindow, 49
- getSourceDirectory
  - AppLogic::ImageLoader, 94
- getSourceFileLocation
  - AppLogic::VecImage, 104
- getVecDoubleData
  - AppLogic::VecImage, 104
- getWorkingImagesMat
  - AppLogic::Filter, 27
- hideListItems
  - uihelp, 19
- image
  - UI::FilterWindow, 57
  - UI::ImageDeconvolutionWindow, 86
- image\_calc\_amount
  - AppLogic::FilterInfo, 35
- image\_set
  - AppLogic::Filter, 31
- ImageCleaningLogic.cpp, 117
- ImageCleaningLogic.hpp, 118
- ImageDeconvolution
  - AppLogic::ImageDeconvolution, 67
- ImageDeconvolution.cpp, 119
- ImageDeconvolution.hpp, 119
  - FILT\_IMAGE\_SUFFIX, 121
- ImageDeconvolutionWindow
  - UI::ImageDeconvolutionWindow, 78
- ImageDeconvolutionWindow.cpp, 121
  - ELIDE\_WIDTH, 122
- ImageDeconvolutionWindow.h, 122
- ImageLoader.cpp, 123
- ImageLoader.hpp, 124
  - FILTER\_FILE\_EXTENSION, 125
  - FILTER\_INFO\_EXTENSION, 125
  - TEMP\_FILE\_NAME, 126
  - ZIP\_FILE\_BUFF\_SIZE, 126
- img\_cleaning
  - UI::ImageDeconvolutionWindow, 86
- importFilter
  - UI::ImageDeconvolutionWindow, 79
- importFilterFromZip
  - AppLogic::ImageLoader, 94
- initializeImageFileDialog
  - uihelp, 19
- is\_canceled
  - AppLogic::Filter, 31
  - AppLogic::FilterLogic, 43
  - AppLogic::ImageDeconvolution, 73
- is\_filter\_loaded
  - UI::ImageDeconvolutionWindow, 87
- isSupportedImageFormat
  - AppLogic::ImageLoader, 94
- LONGER\_MESSAGE\_TIME
  - uihelpers.h, 132
- LOW\_RANK\_APPROX\_ITERATIONS
  - libfrcima.hpp, 129
- label\_dirty\_image
  - UI::FilterWindow, 57
- label\_filtered\_image
  - UI::ImageDeconvolutionWindow, 87
- label\_loaded\_image
  - UI::FilterWindow, 57
  - UI::ImageDeconvolutionWindow, 87
- last\_noise
  - AppLogic::FilterLogic, 43
- last\_noise\_value
  - AppLogic::FilterLogic, 43
- last\_used\_noise\_type
  - AppLogic::Filter, 31
- last\_used\_noise\_value
  - AppLogic::Filter, 32
- libfrcima.cpp, 126
- libfrcima.hpp, 127
  - LOW\_RANK\_APPROX\_ITERATIONS, 129
- loadExistingFilters
  - AppLogic::ImageDeconvolution, 70
- loadFilter
  - AppLogic::ImageCleaningLogic, 63
  - AppLogic::ImageDeconvolution, 70
- loadFilterInfo
  - AppLogic::ImageLoader, 95
- loadFmatrixFromFile
  - AppLogic::Filter, 27
- loadImageList

- UI::FilterWindow, 49
  - UI::ImageDeconvolutionWindow, 80
- loadImagesFromFile
  - UI::FilterWindow, 50
  - UI::ImageDeconvolutionWindow, 80
- loadImagesFromFolder
  - AppLogic::Filter, 28
  - AppLogic::FilterLogic, 41
  - AppLogic::ImageCleaningLogic, 63
  - AppLogic::ImageDeconvolution, 70
  - AppLogic::ImageLoader, 95
  - UI::FilterWindow, 50
  - UI::ImageDeconvolutionWindow, 80
- loadImagesFromZip
  - AppLogic::Filter, 28
  - AppLogic::FilterLogic, 41
  - AppLogic::ImageCleaningLogic, 64
  - AppLogic::ImageDeconvolution, 72
  - AppLogic::ImageLoader, 95
  - UI::FilterWindow, 50
  - UI::ImageDeconvolutionWindow, 81
- loadSingleImage
  - AppLogic::Filter, 28
  - AppLogic::ImageDeconvolution, 72
  - AppLogic::ImageLoader, 97
- loaded\_file\_names
  - AppLogic::Filter, 32
  - AppLogic::ImageDeconvolution, 74
  - AppLogic::ImageLoader, 98
- loaded\_file\_paths
  - AppLogic::Filter, 32
  - AppLogic::ImageDeconvolution, 74
  - AppLogic::ImageLoader, 98
- loaded\_filters
  - AppLogic::ImageDeconvolution, 74
- loadfilter\_future\_watcher
  - UI::ImageDeconvolutionWindow, 87
- low\_rank
  - frcima, 13
- low\_rank\_approx
  - frcima, 14
- main
  - main.cpp, 130
- main.cpp, 129
- main, 130
- new\_filter
  - AppLogic::FilterLogic, 44
- noise\_type
  - AppLogic::FilterInfo, 35
- noise\_type\_t
  - AppLogic, 10
- noise\_value
  - AppLogic::FilterInfo, 35
- not\_loaded\_files
  - AppLogic::ImageLoader, 98
- notAllowedChars
  - UI, 16
- notAllowedSubStrings
  - UI, 16
- numCols
  - AppLogic::VecImage, 104
- numRows
  - AppLogic::VecImage, 105
- on\_actionCreate\_filter\_triggered
  - UI::ImageDeconvolutionWindow, 81
- on\_actionFilter\_image\_triggered
  - UI::FilterWindow, 51
- on\_actionLoadImageFromZip\_triggered
  - UI::FilterWindow, 51
  - UI::ImageDeconvolutionWindow, 81
- on\_btn\_load\_folder\_clicked
  - UI::FilterWindow, 51
  - UI::ImageDeconvolutionWindow, 81
- on\_btn\_load\_image\_clicked
  - UI::ImageDeconvolutionWindow, 81
- on\_btn\_load\_single\_image\_clicked
  - UI::FilterWindow, 51
- on\_btn\_save\_all\_dirtyimg\_clicked
  - UI::FilterWindow, 51
- on\_btn\_save\_dirtyimg\_clicked
  - UI::FilterWindow, 51
- on\_calc\_filter\_btn\_clicked
  - UI::FilterWindow, 51
- on\_comboBox\_filter\_select\_currentIndexChanged
  - UI::ImageDeconvolutionWindow, 81
- on\_exportFilterCanceled
  - UI::ImageDeconvolutionWindow, 82
- on\_exportFilterFinished
  - UI::ImageDeconvolutionWindow, 82
- on\_filter\_name\_input\_editingFinished
  - UI::FilterWindow, 52
- on\_filterAllImagesCanceled
  - UI::ImageDeconvolutionWindow, 82
- on\_filterAllImagesFinished
  - UI::ImageDeconvolutionWindow, 82
- on\_filterCalculationCanceled
  - UI::FilterWindow, 52
- on\_filterCalculationFinished
  - UI::FilterWindow, 52
- on\_filterLoadingFinished
  - UI::ImageDeconvolutionWindow, 82
- on\_importFilterCanceled
  - UI::ImageDeconvolutionWindow, 82
- on\_importFilterFinished
  - UI::ImageDeconvolutionWindow, 82
- on\_lineEdit\_filter\_loaded\_images\_textChanged
  - UI::FilterWindow, 52
  - UI::ImageDeconvolutionWindow, 83
- on\_listWidget\_loaded\_images\_currentRowChanged
  - UI::FilterWindow, 52
  - UI::ImageDeconvolutionWindow, 83
- on\_loadImagesCanceled
  - UI::FilterWindow, 52
  - UI::ImageDeconvolutionWindow, 83
- on\_loadImagesFinished

- UI::FilterWindow, 52
  - UI::ImageDeconvolutionWindow, 83
- on\_next\_loaded\_image\_btn\_clicked
  - UI::FilterWindow, 53
  - UI::ImageDeconvolutionWindow, 83
- on\_noise\_selection\_group\_currentChanged
  - UI::FilterWindow, 53
- on\_previous\_loaded\_image\_btn\_clicked
  - UI::FilterWindow, 53
  - UI::ImageDeconvolutionWindow, 83
- on\_pushButton\_deletefilter\_clicked
  - UI::ImageDeconvolutionWindow, 83
- on\_pushButton\_exportfilter\_clicked
  - UI::ImageDeconvolutionWindow, 84
- on\_pushButton\_filterall\_save\_clicked
  - UI::ImageDeconvolutionWindow, 84
- on\_pushButton\_importfilter\_clicked
  - UI::ImageDeconvolutionWindow, 84
- on\_pushButton\_save\_filterimage\_clicked
  - UI::ImageDeconvolutionWindow, 84
- on\_saveAllDirtyImagesCanceled
  - UI::FilterWindow, 53
- on\_saveAllDirtyImagesFinished
  - UI::FilterWindow, 53
- on\_slider\_gaussiannoise\_valueChanged
  - UI::FilterWindow, 53
- on\_slider\_riciannoise\_valueChanged
  - UI::FilterWindow, 53
- on\_slider\_snnoise\_valueChanged
  - UI::FilterWindow, 54
- on\_slider\_uniformnoise\_valueChanged
  - UI::FilterWindow, 54
- operator<<
  - AppLogic, 10
- operator>>
  - AppLogic, 11
- operator=
  - AppLogic::VecImage, 105
- progress\_dialog\_calc\_filter
  - UI::FilterWindow, 57
- progress\_dialog\_filter
  - UI::ImageDeconvolutionWindow, 87
- pseudo\_inverse\_tpm
  - frcima, 14
- rank
  - AppLogic::FilterInfo, 35
- rcima
  - frcima, 15
- rows
  - AppLogic::VecImage, 106
- SLASHES
  - FilterWindow.cpp, 115
- STANDARD\_MESSAGE\_TIME
  - uihelpers.h, 132
- save
  - AppLogic::VecImage, 105
- saveAllDirtyImages
  - AppLogic::Filter, 29
  - AppLogic::FilterLogic, 41
  - UI::FilterWindow, 54
- saveAllFilteredImages
  - AppLogic::ImageCleaningLogic, 64
  - AppLogic::ImageDeconvolution, 72
  - UI::ImageDeconvolutionWindow, 84
- saveDirtyImage
  - AppLogic::Filter, 29
  - AppLogic::FilterLogic, 42
  - UI::FilterWindow, 54
- saveFilterInfo
  - AppLogic::Filter, 30
- saveFilteredImage
  - AppLogic::ImageCleaningLogic, 65
  - AppLogic::ImageDeconvolution, 73
  - UI::ImageDeconvolutionWindow, 84
- saveToFile
  - AppLogic::Filter, 30
- selected\_noise\_type
  - UI::FilterWindow, 57
- selected\_noise\_value
  - UI::FilterWindow, 57
- setDirtyImage
  - UI::FilterWindow, 55
- setFilterName
  - AppLogic::Filter, 30
- setFilterNameInputError
  - UI::FilterWindow, 55
- setFilteredImage
  - UI::ImageDeconvolutionWindow, 85
- setImgDataMatDouble
  - AppLogic::VecImage, 106
- setLoadedImage
  - UI::FilterWindow, 55
  - UI::ImageDeconvolutionWindow, 85
- setNoise
  - AppLogic::FilterLogic, 42
- setNoiseValue
  - AppLogic::FilterLogic, 43
- source\_directory
  - AppLogic::ImageLoader, 98
- source\_file\_location
  - AppLogic::VecImage, 106
- stripExtensionFromFilename
  - AppLogic::ImageLoader, 97
- TEMP\_FILE\_NAME
  - ImageLoader.hpp, 126
- UI::FilterWindow, 44
  - ~FilterWindow, 48
  - bool\_future\_watcher, 56
  - createFilterInfoDialog, 49
  - dirty\_image, 56
  - filter\_logic, 56
  - FilterWindow, 48
  - future\_watcher\_calc\_filter, 56



- future\_watcher\_load\_images, 56
- getSelectedNoiseValue, 49
- image, 57
- label\_dirty\_image, 57
- label\_loaded\_image, 57
- loadImageList, 49
- loadImagesFromFile, 50
- loadImagesFromFolder, 50
- loadImagesFromZip, 50
- on\_actionFilter\_image\_triggered, 51
- on\_actionLoadImageFromZip\_triggered, 51
- on\_btn\_load\_folder\_clicked, 51
- on\_btn\_load\_single\_image\_clicked, 51
- on\_btn\_save\_all\_dirtyimg\_clicked, 51
- on\_btn\_save\_dirtyimg\_clicked, 51
- on\_calc\_filter\_btn\_clicked, 51
- on\_filter\_name\_input\_editingFinished, 52
- on\_filterCalculationCanceled, 52
- on\_filterCalculationFinished, 52
- on\_lineEdit\_filter\_loaded\_images\_textChanged, 52
- on\_listWidget\_loaded\_images\_currentRow↔  
Changed, 52
- on\_loadImagesCanceled, 52
- on\_loadImagesFinished, 52
- on\_next\_loaded\_image\_btn\_clicked, 53
- on\_noise\_selection\_group\_currentChanged, 53
- on\_previous\_loaded\_image\_btn\_clicked, 53
- on\_saveAllDirtyImagesCanceled, 53
- on\_saveAllDirtyImagesFinished, 53
- on\_slider\_gaussiannoise\_valueChanged, 53
- on\_slider\_riciannoise\_valueChanged, 53
- on\_slider\_snpnoise\_valueChanged, 54
- on\_slider\_uniformnoise\_valueChanged, 54
- progress\_dialog\_calc\_filter, 57
- saveAllDirtyImages, 54
- saveDirtyImage, 54
- selected\_noise\_type, 57
- selected\_noise\_value, 57
- setDirtyImage, 55
- setFilterNameInputError, 55
- setLoadedImage, 55
- ui, 58
- validateNameInput, 55
- UI::ImageDeconvolutionWindow, 75
  - ~ImageDeconvolutionWindow, 78
  - bool\_future\_watcher, 86
  - displayFilterInfo, 79
  - exportFilter, 79
  - filtered\_image, 86
  - future\_watcher\_load\_images, 86
  - image, 86
  - ImageDeconvolutionWindow, 78
  - img\_cleaning, 86
  - importFilter, 79
  - is\_filter\_loaded, 87
  - label\_filtered\_image, 87
  - label\_loaded\_image, 87
  - loadImageList, 80
  - loadImagesFromFile, 80
  - loadImagesFromFolder, 80
  - loadImagesFromZip, 81
  - loadfilter\_future\_watcher, 87
  - on\_actionCreate\_filter\_triggered, 81
  - on\_actionLoadImageFromZip\_triggered, 81
  - on\_btn\_load\_folder\_clicked, 81
  - on\_btn\_load\_image\_clicked, 81
  - on\_comboBox\_filter\_select\_currentIndexChanged, 81
  - on\_exportFilterCanceled, 82
  - on\_exportFilterFinished, 82
  - on\_filterAllImagesCanceled, 82
  - on\_filterAllImagesFinished, 82
  - on\_filterLoadingFinished, 82
  - on\_importFilterCanceled, 82
  - on\_importFilterFinished, 82
  - on\_lineEdit\_filter\_loaded\_images\_textChanged, 83
  - on\_listWidget\_loaded\_images\_currentRow↔  
Changed, 83
  - on\_loadImagesCanceled, 83
  - on\_loadImagesFinished, 83
  - on\_next\_loaded\_image\_btn\_clicked, 83
  - on\_previous\_loaded\_image\_btn\_clicked, 83
  - on\_pushButton\_deletefilter\_clicked, 83
  - on\_pushButton\_exportfilter\_clicked, 84
  - on\_pushButton\_filterall\_save\_clicked, 84
  - on\_pushButton\_importfilter\_clicked, 84
  - on\_pushButton\_save\_filterimage\_clicked, 84
  - progress\_dialog\_filter, 87
  - saveAllFilteredImages, 84
  - saveFilteredImage, 84
  - setFilteredImage, 85
  - setLoadedImage, 85
  - ui, 87
  - updateFilterList, 86
- UI, 16
  - notAllowedChars, 16
  - notAllowedSubStrings, 16
- Ui, 17
- ui
  - UI::FilterWindow, 58
  - UI::ImageDeconvolutionWindow, 87
- uihelp, 17
  - allSupportedFormatsString, 18
  - file\_dialog\_type\_t, 17
  - filterListItems, 18
  - generateSizeMessage, 18
  - hideListItems, 19
  - initializeImageFileDialog, 19
- uihelpers.h, 130
  - ELEMENT\_NUMBER\_MILLION\_CUTOFF, 132
  - ELEMENT\_NUMBER\_THOUSAND\_CUTOFF, 132
  - LONGER\_MESSAGE\_TIME, 132
  - STANDARD\_MESSAGE\_TIME, 132

- updateFilterList
  - UI::ImageDeconvolutionWindow, [86](#)
- VECIMG\_NORM
  - VecImage.cpp, [133](#)
- validateNameInput
  - UI::FilterWindow, [55](#)
- VecImage
  - AppLogic::VecImage, [101](#), [102](#)
- VecImage.cpp, [133](#)
  - VECIMG\_NORM, [133](#)
- VecImage.hpp, [134](#)
  - cimg\_display, [135](#)
  - cimg\_use\_jpeg, [135](#)
  - cimg\_use\_png, [135](#)
- WINDOWS\_DEVICES
  - FilterWindow.cpp, [115](#)
- working\_images
  - AppLogic::Filter, [32](#)
  - AppLogic::ImageDeconvolution, [74](#)
- ZIP\_FILE\_BUFF\_SIZE
  - ImageLoader.hpp, [126](#)