Wavelet and Image class library

1.2-cvs

Generated by Doxygen 1.5.5

Thu Apr 17 11:47:12 2008

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Module Documentation

6.1 Classes for images

Classes

- class AviReader
- class AviWriter
- class CoeffInformation
- class ColorBuffer
- class ColorImage
- class ColorVideo
- class ColumnVector
- class FileName
- class FullVector
- class GreymapFile
- class GreymapReader
- class GreymapWriter
- class Image
- class ImageArray< T >
- class ImageVector
- class JpgReader
- class JpgWriter
- class PfcReader
- class PfcWriter
- class PfgReader
- class PfgWriter
- class PgmReader
- class PgmWriter
- class PixmapFile
- class PixmapReader
- class PixmapWriter
- class PpmReader

- class PpmWriter
- class RawReader
- class RawWriter
- class ReferenceVector
- class RowVector
- class StillImage
- class VideoArray< T >
- class VideoFile
- class VideoFrame
- class VideoReader
- class VideoWriter
- class VidReader
- class VidWriter

Defines

- #define STRR(X) #X
- #define STR(X) STRR(X)
- #define DPRINTF(X) (debug_printf("DEBUG %s [%s]: ", __FILE__, STR(__LINE__)), debug_printf X)
- #define NEW(a) new a
- #define DELETE(a) delete (a), (a) = NULL
- #define DELETEAR(a) delete [ ] (a), (a) = NULL
- #define DELETENOTNULL(a) if (a) { DELETE (a); } else {}
- #define DELETENOTNULLAR(a) if (a) { DELETEAR (a); } else {}
- #define COEFF_EPSILON ((coeff)DBL_EPSILON)
- #define COEFF_MAX ((coeff)DBL_MAX)
- #define COEFF_MIN ((coeff)DBL_MIN)
- #define MIN(x, y) ((x) < (y)? (x): (y))
- #define MAX(x, y) ((x) > (y)? (x): (y))
- #define DIV2(x) ((unsigned)(x)>>1u)
- #define MUL2(x) ((unsigned)(x)<<1u)
- #define TWOPOW(n) (1u<<(unsigned)(n))
- #define SQUARE(n) ((n)*(n))
- #define FMT0 0

Typedefs

- typedef unsigned char pixel
- typedef double coeff

Enumerations

- enum filetype {
  fn_pgm, fn_raw, fn_pfi, fn_ppm,
  fn_jpg, fn_vid, fn_avi, fn_unknown }
- enum area {
  LL = 0, HL, LH, HH,
  areaINVALID }
- enum clrmodel { cm_rgb, cm_yuv, cm_grey, cm_unknown }
- enum yuv { yuv_y = 0, yuv_u, yuv_v, yuv_unknown }
- enum rgb { rgb_r = 0, rgb_g, rgb_b, rgb_unknown }

Functions

- pixel * ppm_read (const char *fname, int *height, int *width, int *cmax)
- int debug_printf (const char *tmpl,...)
- pixel tools_coeff2pixel (coeff c)
- int tools_coeff2int (coeff c)
- bool tools_equals (double d1, double d2)
- bool tools_epsilons (double d1, double d2, double epsilon)
- bool tools_areaFromString (char a1, char a2, area &result)
- bool tools_areaFromString (char *a, area &result)
- const char * tools_areaToString (const area a)
- bool tools_powerOfTwo (int value, int &power)
- int tools_startFromCenter (int pos, int length)
- int tools_fileSize (const char *fname)

Variables

- const area areas [areaINVALID]

6.1.1 Detailed Description

6.1.2 Define Documentation

6.1.2.1 #define COEFF_EPSILON ((coeff)DBL_EPSILON)

Coeff epsilon. The epsilon for equals-operations on the {*coeff*} type.

Definition at line 51 of file miscdefs.h.

6.1.2.2 #define COEFF_MAX ((coeff)DBL_MAX)

Coeff maximal value.

Definition at line 54 of file miscdefs.h.

6.1.2.3 #define COEFF_MIN ((coeff)DBL_MIN)

Coeff minimal value.

Definition at line 57 of file miscdefs.h.

6.1.2.4 #define DELETE(a) delete (a), (a) = NULL

Definition at line 30 of file miscdefs.h.

Referenced by NTree< CoeffInformation >::destroyAt().

6.1.2.5 #define DELETEAR(a) delete [ ] (a), (a) = NULL

Definition at line 33 of file miscdefs.h.

6.1.2.6 #define DELETENOTNULL(a) if (a) { DELETE (a); } else {}

Definition at line 36 of file miscdefs.h.

6.1.2.7 #define DELETENOTNULLAR(a) if (a) { DELETEAR (a); } else {}

Definition at line 39 of file miscdefs.h.

6.1.2.8 #define DIV2(x) ((unsigned)(x)>>1u)

Macro for dividing a number by two using a shift operation.

Definition at line 66 of file miscdefs.h.

6.1.2.9 #define DPRINTF(X) (debug_printf("DEBUG %s [%s]: ", __FILE__, STR(__LINE__)), debug_printf X)

Definition at line 21 of file debug.h.

6.1.2.10 #define FMT0 0

Macro to circumvent problems with g++-3.0

Definition at line 80 of file miscdefs.h.

6.1.2.11 #define MAX(x, y) ((x) > (y)? (x): (y))

A two-tuple's minimum value.

Definition at line 63 of file miscdefs.h.

6.1.2.12 #define MIN(x, y) ((x) < (y)? (x): (y))

A two-tuple's maximum value.

Definition at line 60 of file miscdefs.h.

6.1.2.13 #define MUL2(x) ((unsigned)(x)<<1u)

Macro for multiplying a number by two using a shift operation.

Definition at line 69 of file miscdefs.h.

6.1.2.14 #define NEW(a) new a

Definition at line 27 of file miscdefs.h.

6.1.2.15 #define SQUARE(n) ((n)*(n))

Definition at line 74 of file miscdefs.h.

6.1.2.16 #define STR(X) STRR(X)

Definition at line 20 of file debug.h.

6.1.2.17 #define STRR(X) #X

Definition at line 19 of file debug.h.

6.1.2.18 #define TWOPOW(n) (1u<<(unsigned)(n))

Macro for calculating pow (2, n) using a shift operation.

Definition at line 72 of file miscdefs.h.

6.1.3 Typedef Documentation

6.1.3.1 typedef double coeff

A coefficient value. A long floating point type to represent one greyscale value for transforms or the pfi image format.

Definition at line 47 of file miscdefs.h.

6.1.3.2 typedef unsigned char pixel

A pixel value. A one-byte integer type to store one greyscale value in image files.

Definition at line 44 of file miscdefs.h.

6.1.4 Enumeration Type Documentation

6.1.4.1 enum area

The four regions of a one-step transform. If there was more than one step performed we'll find the same structure in the {*LL}* part again. The two rows are: {*LL}*, {*HL}*, {*LH}* and {*HH}*.

Enumerator:

LL

HL

LH

HH

areaINVALID

Definition at line 86 of file miscdefs.h.

6.1.4.2 enum clrmodel

Color models. Either RGB or YUV.

Enumerator:

cm_rgb

cm_yuv

cm_grey

cm_unknown

Definition at line 92 of file miscdefs.h.

6.1.4.3 enum filetype

File types. These are tags standing for known and supported file formats or unknown files.

Enumerator:

fn_pgm

fn_raw

fn_pfi

fn_ppm
fn_jpg
fn_vid
fn_avi
fn_unknown

Definition at line 19 of file FileName.hh.

6.1.4.4 enum rgb

RGB color model channel names.

Enumerator:

rgb_r
rgb_g
rgb_b
rgb_unknown

Definition at line 98 of file miscdefs.h.

6.1.4.5 enum yuv

YUV color model channel names.

Enumerator:

yuv_y
yuv_u
yuv_v
yuv_unknown

Definition at line 95 of file miscdefs.h.

6.1.5 Function Documentation

6.1.5.1 int debug_printf (const char * *tmpl*, ...)

6.1.5.2 pixel* ppm_read (const char * *fname*, int * *height*, int * *width*, int * *cmax*)

Read a PPM file from the filesystem.

Parameters:

fname the filename

height the number of pixel rows, returned by the function

width the number of pixel cols, returned by the function

cmax the maximal color value, returned by the function

Returns:

an array containing the pixels as RGB triples if successful else NULL

6.1.5.3 bool tools_areaFromString (char * *a*, area & *result*)

Calculate an area from a string

Parameters:

a the string containing the area name

result the result

Returns:

true if a valid area name was passed, else false

6.1.5.4 bool tools_areaFromString (char *a1*, char *a2*, area & *result*)

Calculate an area from a string split to two characters

Parameters:

a1 the first character

a2 the second character

result the result

Returns:

true if a valid area name was passed, else false

6.1.5.5 const char* tools_areaToString (const area *a*)

Return an external string constant containing the name of the area passed to the function.

Parameters:

a the area

Returns:

the area's name

6.1.5.6 int tools_coeff2int (coeff *c*)

Convert a coeff value to a integer

Parameters:

c the coefficient

Returns:

the integer value

6.1.5.7 pixel tools_coeff2pixel (coeff *c*)

Convert a coeff value to a pixel.

Parameters:

c the coefficient

Returns:

the pixel value

6.1.5.8 bool tools_epsilons (double *d1*, double *d2*, double *epsilon*)

Return true if two double values are almost equal

Parameters:

d1 the first value

d2 the second value

epsilon the comparison epsilon

Returns:

true if almost equal

6.1.5.9 bool tools_equals (double *d1*, double *d2*)

Return true if two double values are equal

Parameters:

d1 the first value

d2 the second value

Returns:

true if equal

6.1.5.10 int tools_fileSize (const char * *fname*)

Return the size of a file in bytes.

Exceptions:

invalid_argument if the file cannot be opened

Parameters:

fname the file name

Returns:

the size in bytes.

6.1.5.11 bool tools_powerOfTwo (int *value*, int & *power*)

Find out whether a number is a power of two.

Parameters:

value the value to test

power (returned value) the power of two if the result is true.

Returns:

true if value is a power of two.

6.1.5.12 int tools_startFromCenter (int *pos*, int *length*)

Calculate a vector's start index so that a given position is in its center.

Parameters:

pos the position

length the vector's length

Returns:

the start index

6.1.6 Variable Documentation

6.1.6.1 const area areas[areaINVALID]

Array for easier iteration over the areas.

6.2 Classes for Wavelet Transforms

Classes

- class Filter
- class FilterSet
- class MirrorPosition
- class PeriodicPosition
- class PyramidTransform
- class StandardTransform
- class VectorPosition
- class Wavelet
- class WaveletTransform

Defines

- #define NULL 0

Variables

- FilterSet Haar
- FilterSet Daub4
- FilterSet Daub6
- FilterSet Daub8
- FilterSet Antonini
- FilterSet Brislawn
- FilterSet Villa1
- FilterSet Villa2
- FilterSet Villa3
- FilterSet Villa4
- FilterSet Villa5
- FilterSet Villa6
- FilterSet Odegard

6.2.1 Detailed Description

Originally this was wavelet.hh from Geoff Davis' Wavelet construction kit. This version has been adapted to my classes resulting in small changes to names and formatting etc. I also removed the class Wavelet which looks different in my code!

Date

2005/07/12 14:52:20

Revision

1.3

6.2.2 Define Documentation

6.2.2.1 #define NULL 0

Definition at line 35 of file Filter.hh.

Referenced by NTree< CoeffInformation >::childAt(), Filter::Filter(), FilterSet::FilterSet(), NTree< CoeffInformation >::hasChildAt(), and NTree< CoeffInformation >::isRoot().

6.2.3 Variable Documentation

6.2.3.1 FilterSet Antonini

6.2.3.2 FilterSet Brislawn

6.2.3.3 FilterSet Daub4

6.2.3.4 FilterSet Daub6

6.2.3.5 FilterSet Daub8

6.2.3.6 FilterSet Haar

6.2.3.7 FilterSet Odegard

6.2.3.8 FilterSet Villa1

6.2.3.9 FilterSet Villa2

6.2.3.10 FilterSet Villa3

6.2.3.11 FilterSet Villa4

6.2.3.12 FilterSet Villa5

6.2.3.13 FilterSet Villa6

6.3 Utilities for images and Wavelet Transforms

Classes

- class Histogram
- struct lq
- struct logvals
- class ImageComparison
- class ImageDenoiser
- class ImageInformation
- class ImageResizer
- class NTree< Type >
- class PyramidTree

Defines

- #define DENOISE_HL 0x01
- #define DENOISE_LH 0x02
- #define DENOISE_HH 0x04
- #define SIGNIFICANT_COEFF 0
- #define SIGNIFICANT_REGION 1
- #define SIGNIFICANT_CHANNEL 2
- #define REPLACE_SIMPLE 0
- #define REPLACE_CHANNEL 1
- #define PII_YPOS(info, pos) (((info) → at (pos)).ypos ())
- #define PII_XPOS(info, pos) (((info) → at (pos)).xpos ())
- #define PII_XYPOS(info, pos) (((info) → at (pos)).xypos ())
- #define II_YPOS(info, pos) (((info).at (pos)).ypos ())
- #define II_XPOS(info, pos) (((info).at (pos)).xpos ())
- #define II_XYPOS(info, pos) (((info).at (pos)).xypos ())

Typedefs

- typedef bool(∗ cipredicate)(const CoeffInformation ∗c1, const CoeffInformation ∗c2)

Enumerations

- enum imgtype { DRAWN = 0, SCANNED }

6.3.1 Detailed Description

6.3.2 Define Documentation

6.3.2.1 #define DENOISE_HH 0x04

Include the HH area in the denoising process

Definition at line 24 of file ImageDenoiser.hh.

6.3.2.2 #define DENOISE_HL 0x01

Include the HL area in the denoising process

Definition at line 20 of file ImageDenoiser.hh.

6.3.2.3 #define DENOISE_LH 0x02

Include the LH area in the denoising process

Definition at line 22 of file ImageDenoiser.hh.

6.3.2.4 #define II_XPOS(info, pos) (((info).at (pos)).xpos ())

Definition at line 27 of file ImageInformation.hh.

6.3.2.5 #define II_XYPOS(info, pos) (((info).at (pos)).xypos ())

Definition at line 28 of file ImageInformation.hh.

6.3.2.6 #define II_YPOS(info, pos) (((info).at (pos)).ypos ())

Definition at line 26 of file ImageInformation.hh.

6.3.2.7 #define PII_XPOS(info, pos) (((info) → at (pos)).xpos ())

Definition at line 24 of file ImageInformation.hh.

6.3.2.8 #define PII_XYPOS(info, pos) (((info) → at (pos)).xypos ())

Definition at line 25 of file ImageInformation.hh.

6.3.2.9 #define PII_YPOS(info, pos) (((info) → at (pos)).ypos ())

Definition at line 23 of file ImageInformation.hh.

6.3.2.10 #define REPLACE_CHANNEL 1

Definition at line 31 of file ImageDenoiser.hh.

6.3.2.11 #define REPLACE_SIMPLE 0

Definition at line 30 of file ImageDenoiser.hh.

6.3.2.12 #define SIGNIFICANT_CHANNEL 2

Definition at line 28 of file ImageDenoiser.hh.

6.3.2.13 #define SIGNIFICANT_COEFF 0

Definition at line 26 of file ImageDenoiser.hh.

6.3.2.14 #define SIGNIFICANT_REGION 1

Definition at line 27 of file ImageDenoiser.hh.

6.3.3 Typedef Documentation

6.3.3.1 typedef bool(∗ cipredicate)(const CoeffInformation ∗c1, const CoeffInformation ∗c2)

A comparison function type for coefficients. Different criterions (e.g. value or index) can be used.

Definition at line 21 of file ImageInformation.hh.

6.3.4 Enumeration Type Documentation

6.3.4.1 enum imgtype

The type of the image. Either hand-drawn or scanned.

Enumerator:

DRAWN

SCANNED

Definition at line 20 of file ImageComparison.hh.

Chapter 7

Namespace Documentation

7.1 MagickInter Namespace Reference

Functions

- Magick::Image magickImageFromColorImageWithTransparency (ColorImage &img, bool withTransparency=false, coeff *transparentColors=NULL, int colorBytes=1)
- Magick::Image magickImageFromColorImage (ColorImage &img)
- std::auto_ptr< ColorImage > colorImageFromMagickImage (Magick::Image &img)
- std::auto_ptr< ColorImage > obtainColorImage (const std::string &inFile)
- void writeColorImage (ColorImage &img, const std::string &outFile, int quality=100)
- void writeColorImageWithTransparency (ColorImage &img, const std::string &outFile, int quality=100, bool withTransparency=false, coeff *transparentColors=NULL, int colorBytes=1)
- void scaleAndWriteColorImage (ColorImage &img, int rows, int cols, const std::string &outFile, int quality=100)

7.1.1 Detailed Description

Helper functions for extended read/write support using the Magick++ classes. This is motivated by the fact that we want to support many, many file formats but don't want to spend our time writing import and export filters. The implementation is quite rudimentary. All conversion is done by saving and loading temporary files. Also, naturally (as the Wavelet lib does not have support for this), all meta-information like e.g. transparency is lost.

7.1.2 Function Documentation

7.1.2.1 std::auto_ptr<ColorImage> MagickInter::colorImageFromMagickImage (Magick::Image & *img*)

Converts a Magick::Image to a ColorImage.

Parameters:

img the Magick::Image object

Returns:

an auto_ptr to the corresponding ColorImage

Exceptions:

ios_base:failure if an I/O operation did not succeed (e.g. non-existent file, insufficient disk space etc.)

invalid_argument for logical errors (e.g. unknown image file format etc.)

7.1.2.2 Magick::Image MagickInter::magickImageFromColorImage (ColorImage & *img*)

Converts a ColorImage to a Magick::Image.

Parameters:

img the ColorImage object

Returns:

a corresponding Magick::Image

Exceptions:

ios_base:failure if an I/O operation did not succeed (e.g. non-existent file, insufficient disk space etc.)

invalid_argument for logical errors (e.g. unknown image file format etc.)

7.1.2.3 Magick::Image MagickInter::magickImageFromColorImageWithTransparency (ColorImage & *img*, bool *withTransparency* = `false`, coeff * *transparentColors* = `NULL`, int *colorBytes* = `1`)

Converts a ColorImage to a Magick::Image.

Parameters:

img the ColorImage object

withTransparency if set to true, the returned image will contain transparency information according to the original image's colors. NOTE: this will only work with 3-color images using RGB color model.

transparentColors an array of as many members as the image has color channels. If at a given position the original image's pixel n-tuple has identical values, the pixel will be considered transparent. The default value NULL stands for { -1, -1, -1 }

colorBytes the number of bytes per pixel (usually 1, i.e. 256 distinct colors)

Returns:

a corresponding Magick::Image

Exceptions:

ios_base:failure if an I/O operation did not succeed (e.g. non-existent file, insufficient disk space etc.)

invalid_argument for logical errors (e.g. unknown image file format etc.)

7.1.2.4 std::auto_ptr<ColorImage> MagickInter::obtainColorImage (const std::string & *inFile*)

Creates a ColorImage object for an image in the file system using the Magick classes.

Parameters:

inFile the input file name

Returns:

an auto_ptr to the new ColorImage

Exceptions:

ios_base:failure if an I/O operation did not succeed (e.g. non-existent file, insufficient disk space etc.)

invalid_argument for logical errors (e.g. unknown image file format etc.)

7.1.2.5 void MagickInter::scaleAndWriteColorImage (ColorImage & *img*, int *rows*, int *cols*, const std::string & *outFile*, int *quality* = 100)

Rescales and then writes a ColorImage to any file format using the Magick classes.

Parameters:

img the ColorImage object

rows the target number of image rows

cols the target number of image cols

outFile the output file name

quality the quality (e.g. JPEG quality) if applicable

Exceptions:

ios_base:failure if an I/O operation did not succeed (e.g. non-existent file, insufficient disk space etc.)

invalid_argument for logical errors (e.g. unknown image file format etc.)

7.1.2.6 void MagickInter::writeColorImage (ColorImage & *img*, const std::string & *outFile*, int *quality* = 100)

Writes a ColorImage to any file format using the Magick classes.

Parameters:

img the ColorImage object

outFile the output file name

quality the quality (e.g. JPEG quality) if applicable

Exceptions:

ios_base:failure if an I/O operation did not succeed (e.g. non-existent file, insufficient disk space etc.)

invalid_argument for logical errors (e.g. unknown image file format etc.)

7.1.2.7 void MagickInter::writeColorImageWithTransparency (ColorImage & *img*, const std::string & *outFile*, int *quality* = 100, bool *withTransparency* = false, coeff * *transparentColors* = NULL, int *colorBytes* = 1)

Writes a ColorImage to any file format using the Magick classes. This does not use the usual export/import approach but creates an intermediate Magick image from the scratch. However it should be noted that the result may differ slightly from the source image due to Magick's quantization.

Parameters:

img the ColorImage object

outFile the output file name

quality the quality (e.g. JPEG quality) if applicable

withTransparency if set to true, the returned image will contain transparency information according to the original image's colors. NOTE: this will only work with 3-color images using RGB color model.

transparentColors an array of as many members as the image has color channels. If at a given position the original image's pixel n-tuple has identical values, the pixel will be considered transparent. The default value NULL stands for { -1, -1, -1 }

colorBytes the number of bytes per pixel (usually 1, i.e. 256 distinct colors)

Exceptions:

ios_base:failure if an I/O operation did not succeed (e.g. non-existent file, insufficient disk space etc.)

invalid_argument for logical errors (e.g. unknown image file format etc.)

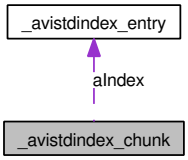
Chapter 8

Class Documentation

8.1 _avistdindex_chunk Struct Reference

`#include <avilib.h>`

Collaboration diagram for _avistdindex_chunk:



Public Attributes

- char fcc [4]
- uint32_t dwSize
- uint16_t wLongsPerEntry
- uint8_t bIndexSubType
- uint8_t bIndexType
- uint32_t nEntriesInUse
- char dwChunkId [4]
- uint64_t qwBaseOffset
- uint32_t dwReserved3
- avistdindex_entry * aIndex

8.1.1 Detailed Description

Definition at line 162 of file avilib.h.

8.1.2 Member Data Documentation

8.1.2.1 char _avistdindex_chunk::fcc[4]

Definition at line 163 of file avilib.h.

8.1.2.2 uint32_t _avistdindex_chunk::dwSize

Definition at line 164 of file avilib.h.

8.1.2.3 uint16_t _avistdindex_chunk::wLongsPerEntry

Definition at line 165 of file avilib.h.

8.1.2.4 uint8_t _avistdindex_chunk::bIndexSubType

Definition at line 166 of file avilib.h.

8.1.2.5 uint8_t _avistdindex_chunk::bIndexType

Definition at line 167 of file avilib.h.

8.1.2.6 uint32_t _avistdindex_chunk::nEntriesInUse

Definition at line 168 of file avilib.h.

8.1.2.7 char _avistdindex_chunk::dwChunkId[4]

Definition at line 169 of file avilib.h.

8.1.2.8 uint64_t _avistdindex_chunk::qwBaseOffset

Definition at line 170 of file avilib.h.

8.1.2.9 uint32_t _avistdindex_chunk::dwReserved3

Definition at line 171 of file avilib.h.

8.1.2.10 avistdindex_entry* _avistdindex_chunk::aIndex

Definition at line 172 of file avilib.h.

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

- avilib.h

8.2 _avistdindex_entry Struct Reference

`#include <avilib.h>`

Public Attributes

- uint32_t dwOffset
- uint32_t dwSize

8.2.1 Detailed Description

Definition at line 156 of file avilib.h.

8.2.2 Member Data Documentation

8.2.2.1 uint32_t _avistdindex_entry::dwOffset

Definition at line 157 of file avilib.h.

8.2.2.2 uint32_t _avistdindex_entry::dwSize

Definition at line 158 of file avilib.h.

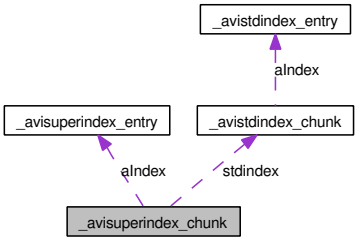
The documentation for this struct was generated from the following file:

- avilib.h

8.3 _avisuperindex_chunk Struct Reference

`#include <avilib.h>`

Collaboration diagram for _avisuperindex_chunk:



Public Attributes

- char fcc [4]
- uint32_t dwSize
- uint16_t wLongsPerEntry
- uint8_t bIndexSubType
- uint8_t bIndexType
- uint32_t nEntriesInUse
- char dwChunkId [4]
- uint32_t dwReserved [3]
- avisuperindex_entry * aIndex
- avistdindex_chunk ** stdindex

8.3.1 Detailed Description

Definition at line 177 of file avilib.h.

8.3.2 Member Data Documentation

8.3.2.1 char _avisuperindex_chunk::fcc[4]

Definition at line 178 of file avilib.h.

8.3.2.2 uint32_t _avisuperindex_chunk::dwSize

Definition at line 179 of file avilib.h.

8.3.2.3 uint16_t _avisuperindex_chunk::wLongsPerEntry

Definition at line 180 of file avilib.h.

8.3.2.4 uint8_t _avisuperindex_chunk::bIndexSubType

Definition at line 181 of file avilib.h.

8.3.2.5 uint8_t _avisuperindex_chunk::bIndexType

Definition at line 182 of file avilib.h.

8.3.2.6 uint32_t _avisuperindex_chunk::nEntriesInUse

Definition at line 183 of file avilib.h.

8.3.2.7 char _avisuperindex_chunk::dwChunkId[4]

Definition at line 184 of file avilib.h.

8.3.2.8 uint32_t _avisuperindex_chunk::dwReserved[3]

Definition at line 185 of file avilib.h.

8.3.2.9 avisuperindex_entry* _avisuperindex_chunk::aIndex

Definition at line 187 of file avilib.h.

8.3.2.10 avistdindex_chunk** _avisuperindex_chunk::stdindex

Definition at line 188 of file avilib.h.

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

- avilib.h

8.4 _avisuperindex_entry Struct Reference

`#include <avilib.h>`

Public Attributes

- uint64_t qwOffset
- uint32_t dwSize
- uint32_t dwDuration

8.4.1 Detailed Description

Definition at line 150 of file avilib.h.

8.4.2 Member Data Documentation

8.4.2.1 uint64_t _avisuperindex_entry::qwOffset

Definition at line 151 of file avilib.h.

8.4.2.2 uint32_t _avisuperindex_entry::dwSize

Definition at line 152 of file avilib.h.

8.4.2.3 uint32_t _avisuperindex_entry::dwDuration

Definition at line 153 of file avilib.h.

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

- avilib.h

8.5 alBITMAPINFOHEADER Struct Reference

`#include <avilib.h>`

Public Attributes

- uint32_t bi_size
- uint32_t bi_width
- uint32_t bi_height
- uint16_t bi_planes
- uint16_t bi_bit_count
- uint32_t bi_compression
- uint32_t bi_size_image
- uint32_t bi_x_pels_per_meter
- uint32_t bi_y_pels_per_meter
- uint32_t bi_clr_used
- uint32_t bi_clr_important

8.5.1 Detailed Description

Definition at line 220 of file avilib.h.

8.5.2 Member Data Documentation

8.5.2.1 uint32_t alBITMAPINFOHEADER::bi_size

Definition at line 222 of file avilib.h.

8.5.2.2 uint32_t alBITMAPINFOHEADER::bi_width

Definition at line 223 of file avilib.h.

8.5.2.3 uint32_t alBITMAPINFOHEADER::bi_height

Definition at line 224 of file avilib.h.

8.5.2.4 uint16_t alBITMAPINFOHEADER::bi_planes

Definition at line 225 of file avilib.h.

8.5.2.5 uint16_t alBITMAPINFOHEADER::bi_bit_count

Definition at line 226 of file avilib.h.

8.5.2.6 uint32_t alBITMAPINFOHEADER::bi_compression

Definition at line 227 of file avilib.h.

8.5.2.7 uint32_t alBITMAPINFOHEADER::bi_size_image

Definition at line 228 of file avilib.h.

8.5.2.8 uint32_t alBITMAPINFOHEADER::bi_x_pels_per_meter

Definition at line 229 of file avilib.h.

8.5.2.9 uint32_t alBITMAPINFOHEADER::bi_y_pels_per_meter

Definition at line 230 of file avilib.h.

8.5.2.10 uint32_t alBITMAPINFOHEADER::bi_clr_used

Definition at line 231 of file avilib.h.

8.5.2.11 uint32_t alBITMAPINFOHEADER::bi_clr_important

Definition at line 232 of file avilib.h.

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

- avilib.h

8.6 audio_index_entry Struct Reference

`#include <avilib.h>`

Public Attributes

- off_t pos
- off_t len
- off_t tot

8.6.1 Detailed Description

Definition at line 126 of file avilib.h.

8.6.2 Member Data Documentation

8.6.2.1 off_t audio_index_entry::pos

Definition at line 128 of file avilib.h.

8.6.2.2 off_t audio_index_entry::len

Definition at line 129 of file avilib.h.

8.6.2.3 off_t audio_index_entry::tot

Definition at line 130 of file avilib.h.

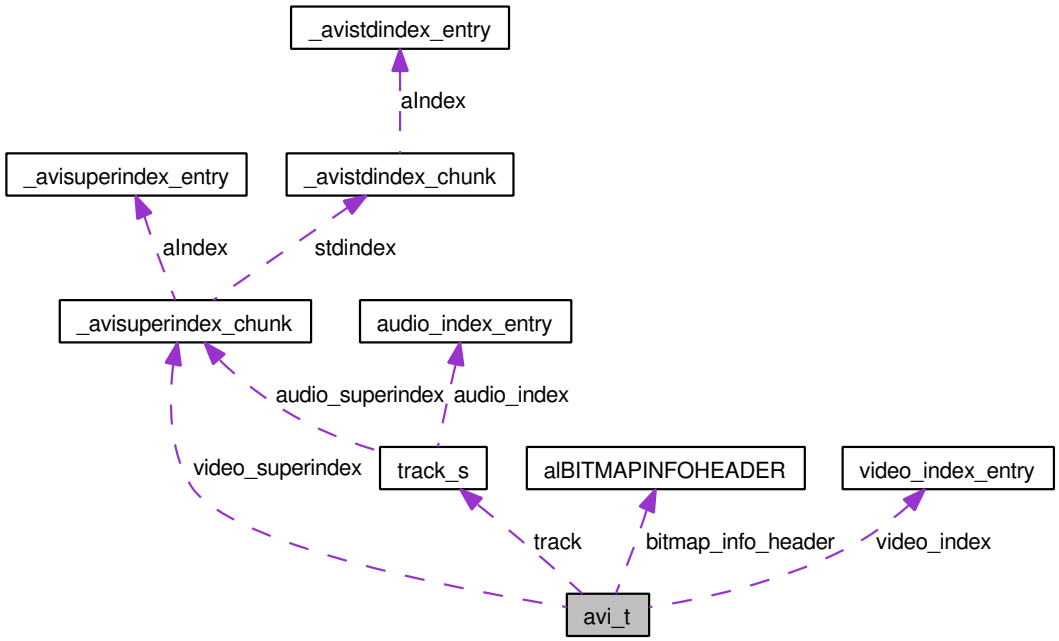
The documentation for this struct was generated from the following file:

- avilib.h

8.7 avi_t Struct Reference

`#include <avilib.h>`

Collaboration diagram for avi_t:



Public Attributes

- long fdes
- long mode
- long width
- long height
- double fps
- char compressor [8]
- char compressor2 [8]
- long video_strn
- long video_frames
- char video_tag [4]
- long video_pos
- uint32_t max_len
- track_t track [AVI_MAX_TRACKS]
- off_t pos
- long n_idx
- long max_idx
- off_t v_codech_off

- off_t v_codecf_off
- uint8_t(* idx)[16]
- video_index_entry * video_index
- avisuperindex_chunk * video_superindex
- int is_opendml
- off_t last_pos
- uint32_t last_len
- int must_use_index
- off_t movi_start
- int total_frames
- int anum
- int aptr
- int comment_fd
- char * index_file
- alBITMAPINFOHEADER * bitmap_info_header
- alWAVEFORMATEX * wave_format_ex [AVI_MAX_TRACKS]

8.7.1 Detailed Description

Definition at line 271 of file avilib.h.

8.7.2 Member Data Documentation

8.7.2.1 long avi_t::fdes

Definition at line 274 of file avilib.h.

8.7.2.2 long avi_t::mode

Definition at line 275 of file avilib.h.

8.7.2.3 long avi_t::width

Definition at line 277 of file avilib.h.

8.7.2.4 long avi_t::height

Definition at line 278 of file avilib.h.

8.7.2.5 double avi_t::fps

Definition at line 279 of file avilib.h.

8.7.2.6 char avi_t::compressor[8]

Definition at line 280 of file avilib.h.

8.7.2.7 char avi_t::compressor2[8]

Definition at line 281 of file avilib.h.

8.7.2.8 long avi_t::video_strn

Definition at line 282 of file avilib.h.

8.7.2.9 long avi_t::video_frames

Definition at line 283 of file avilib.h.

8.7.2.10 char avi_t::video_tag[4]

Definition at line 284 of file avilib.h.

8.7.2.11 long avi_t::video_pos

Definition at line 285 of file avilib.h.

8.7.2.12 uint32_t avi_t::max_len

Definition at line 288 of file avilib.h.

8.7.2.13 track_t avi_t::track[AVI_MAX_TRACKS]

Definition at line 290 of file avilib.h.

8.7.2.14 off_t avi_t::pos

Definition at line 292 of file avilib.h.

8.7.2.15 long avi_t::n_idx

Definition at line 293 of file avilib.h.

8.7.2.16 long avi_t::max_idx

Definition at line 294 of file avilib.h.

8.7.2.17 off_t avi_t::v_codech_off

Definition at line 296 of file avilib.h.

8.7.2.18 off_t avi_t::v_codecf_off

Definition at line 297 of file avilib.h.

8.7.2.19 uint8_t(* avi_t::idx)[16]

Definition at line 299 of file avilib.h.

8.7.2.20 video_index_entry* avi_t::video_index

Definition at line 301 of file avilib.h.

8.7.2.21 avisuperindex_chunk* avi_t::video_superindex

Definition at line 302 of file avilib.h.

8.7.2.22 int avi_t::is_opendml

Definition at line 303 of file avilib.h.

8.7.2.23 off_t avi_t::last_pos

Definition at line 305 of file avilib.h.

8.7.2.24 uint32_t avi_t::last_len

Definition at line 306 of file avilib.h.

8.7.2.25 int avi_t::must_use_index

Definition at line 307 of file avilib.h.

8.7.2.26 off_t avi_t::movi_start

Definition at line 308 of file avilib.h.

8.7.2.27 int avi_t::total_frames

Definition at line 309 of file avilib.h.

8.7.2.28 int avi_t::anum

Definition at line 311 of file avilib.h.

8.7.2.29 int avi_t::aptr

Definition at line 312 of file avilib.h.

8.7.2.30 int avi_t::comment_fd

Definition at line 313 of file avilib.h.

8.7.2.31 char* avi_t::index_file

Definition at line 314 of file avilib.h.

8.7.2.32 alBITMAPINFOHEADER* avi_t::bitmap_info_header

Definition at line 316 of file avilib.h.

8.7.2.33 alWAVEFORMATEX* avi_t::wave_format_ex[AVI_MAX_TRACKS]

Definition at line 317 of file avilib.h.

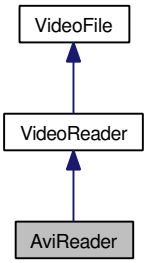
The documentation for this struct was generated from the following file:

- avilib.h

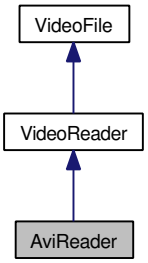
8.8 AviReader Class Reference

`#include <AviReader.hh>`

Inheritance diagram for AviReader:



Collaboration diagram for AviReader:



Public Member Functions

- AviReader (char const *name, VideoArray< coeff > *arrays[ ], int rawy, int rawx, int from, int to)

Static Public Member Functions

- static double frameRate (const char *fname)
- static int framesInFile (const char *fname)
- static void fileDimensions (const char *fname, int &y, int &x)

Protected Member Functions

- virtual int readfmt (void)

Protected Attributes

- int m_ysize

- int m_xsize

8.8.1 Detailed Description

An AVI file reader. Only uncompressed AVIs are supported.

Definition at line 21 of file AviReader.hh.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 AviReader::AviReader (char const * *name*, VideoArray< coeff > * *arrays*[ ], int *rawy*, int *rawx*, int *from*, int *to*)

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference. All frames in the ranges must be of the same size.

Parameters:

name the file name

arrays the array of video arrays for the values

rawy the number of rows

rawx the number of cols

from one greater than the last frame, if equal to 'from' then the whole video will be read, if 0, then the video will be read from 'from' to the end

to the last frame

8.8.3 Member Function Documentation

8.8.3.1 static double AviReader::frameRate (const char * *fname*) [static]

Calculates the frame rate.

Exceptions:

invalid_argument if the file cannot be read

Parameters:

fname the file name

Returns:

the frame rate

8.8.3.2 static int AviReader::framesInFile (const char * *fname*) [static]

Calculates the number of frames in a file.

Exceptions:

invalid_argument if the file cannot be read

Parameters:

fname the file name

Returns:

the number of frames

8.8.3.3 static void AviReader::fileDimensions (const char * *fname*, int & *y*, int & *x*) [static]

Calculates the frame dimensions in a file.

Exceptions:

invalid_argument if the file cannot be read

Parameters:

fname the file name

y the returned number of rows

x the returned number of columns

8.8.3.4 virtual int AviReader::readfmt (void) [protected, virtual]

Read the VID file format. This does the actual work of reading and parsing the image file. It gets called by the {*read* ()} method.

Returns:

0 if successful, -1 on read error, -2 on file format error.

Implements VideoReader.

8.8.4 Member Data Documentation

8.8.4.1 int AviReader::m_ysize [protected]

The assumed number of rows

Definition at line 65 of file AviReader.hh.

8.8.4.2 int AviReader::m_xsize [protected]

The assumed number of columns

Definition at line 67 of file AviReader.hh.

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

- AviReader.hh

8.9 AVIStreamHeader Struct Reference

`#include <avilib.h>`

Public Attributes

- long fccType
- long fccHandler
- long dwFlags
- long dwPriority
- long dwInitialFrames
- long dwScale
- long dwRate
- long dwStart
- long dwLength
- long dwSuggestedBufferSize
- long dwQuality
- long dwSampleSize

8.9.1 Detailed Description

Definition at line 510 of file avilib.h.

8.9.2 Member Data Documentation

8.9.2.1 long AVIStreamHeader::fccType

Definition at line 511 of file avilib.h.

8.9.2.2 long AVIStreamHeader::fccHandler

Definition at line 512 of file avilib.h.

8.9.2.3 long AVIStreamHeader::dwFlags

Definition at line 513 of file avilib.h.

8.9.2.4 long AVIStreamHeader::dwPriority

Definition at line 514 of file avilib.h.

8.9.2.5 long AVIStreamHeader::dwInitialFrames

Definition at line 515 of file avilib.h.

8.9.2.6 long AVIStreamHeader::dwScale

Definition at line 516 of file avilib.h.

8.9.2.7 long AVIStreamHeader::dwRate

Definition at line 517 of file avilib.h.

8.9.2.8 long AVIStreamHeader::dwStart

Definition at line 518 of file avilib.h.

8.9.2.9 long AVIStreamHeader::dwLength

Definition at line 519 of file avilib.h.

8.9.2.10 long AVIStreamHeader::dwSuggestedBufferSize

Definition at line 520 of file avilib.h.

8.9.2.11 long AVIStreamHeader::dwQuality

Definition at line 521 of file avilib.h.

8.9.2.12 long AVIStreamHeader::dwSampleSize

Definition at line 522 of file avilib.h.

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

- avilib.h

8.10 AviWriter Class Reference

`#include <AviWriter.hh>`

Inheritance diagram for AviWriter:

Collaboration diagram for AviWriter:

Public Member Functions

- AviWriter (char const *name, VideoArray< coeff > *arrays[ ], int channels, double fps)

Protected Member Functions

- virtual int writefmt (void)

Protected Attributes

- double m_frameRate

8.10.1 Detailed Description

A AVI image file writer.

Definition at line 21 of file AviWriter.hh.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 AviWriter::AviWriter (char const * *name*, VideoArray< coeff > * *arrays*[ ], int *channels*, double *fps*) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference and set the offset if necessary.

Parameters:

name the file name

arrays the array of video arrays for the values

channels the number of color channels

fps the frame rate per second

Definition at line 36 of file AviWriter.hh.

References m_frameRate.

8.10.3 Member Function Documentation

8.10.3.1 virtual int AviWriter::writefmt (void) [protected, virtual]

Write a AVI video file. This is the method that does the actual work. It will be called by the {*write* ()} method.

Returns:

0 if successful, -1 on write error.

Implements VideoWriter.

8.10.4 Member Data Documentation

8.10.4.1 double AviWriter::m_frameRate [protected]

The frame rate

Definition at line 47 of file AviWriter.hh.

Referenced by AviWriter().

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

- AviWriter.hh

8.11 chunk_struct Struct Reference

`#include <avilib.h>`

Public Attributes

- uint8_t id [4]
- uint32_t len

8.11.1 Detailed Description

Definition at line 479 of file avilib.h.

8.11.2 Member Data Documentation

8.11.2.1 uint8_t chunk_struct::id[4]

Definition at line 481 of file avilib.h.

8.11.2.2 uint32_t chunk_struct::len

Definition at line 482 of file avilib.h.

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

- avilib.h

8.12 CoeffInformation Class Reference

`#include <CoeffInformation.hh>`

Public Member Functions

- CoeffInformation (coeff val=0.0, int y=0, int x=0, int abs=0, long id=-1, coeff misc=0.0)
- CoeffInformation (const CoeffInformation &c)
- ~CoeffInformation (void)
- coeff val (void) const
- int ypos (void) const
- int xpos (void) const
- int id (void) const
- void id (long id)
- void misc (coeff misc)
- coeff misc (void) const
- int xypos (void) const
- void val (coeff val)
- void set (coeff val, int y, int x, int abs, long id=-1, coeff misc=0.0)
- void set (const CoeffInformation &c)
- void set (const Image &img, int ypos, int xpos)
- int socmp (const CoeffInformation &c) const
- int aocmp (const CoeffInformation &c) const
- int pcmp (const CoeffInformation &c) const
- int svcmp (const CoeffInformation &c) const
- int avcmp (const CoeffInformation &c) const
- bool equals (const CoeffInformation &c) const
- void dump (const char *delim=" ", const char *file="") const
- bool operator== (const CoeffInformation &c) const
- bool operator!= (const CoeffInformation &c) const
- CoeffInformation & operator= (const CoeffInformation &c)

Private Attributes

- coeff m_value
- int m_yposition
- int m_xposition
- int m_xyposition
- long m_id
- coeff m_misc

8.12.1 Detailed Description

Coefficient information. The information consists of value, x-offset, y-offset and absolute offset. This is used for statistics and to obtain alternative image representations.

Definition at line 23 of file CoeffInformation.hh.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 CoeffInformation::CoeffInformation (coeff *val* = 0.0, int *y* = 0, int *x* = 0, int *abs* = 0, long *id* = -1, coeff *misc* = 0.0)

Constructor. Creates a new Coefficient.

Parameters:

val the coeff's value

y the coeff's row

x the coeff's col

abs the coeff's absolute offset

id the coeff's unique id

misc for additional information

8.12.2.2 CoeffInformation::CoeffInformation (const CoeffInformation & *c*) [inline]

Copy Constructor.

Parameters:

c the object to copy from

Definition at line 43 of file CoeffInformation.hh.

8.12.2.3 CoeffInformation::~CoeffInformation (void) [inline]

Definition at line 45 of file CoeffInformation.hh.

8.12.3 Member Function Documentation

8.12.3.1 coeff CoeffInformation::val (void) const [inline]

Get the value.

Returns:

the coeff's value

Definition at line 49 of file CoeffInformation.hh.

References m_value.

8.12.3.2 int CoeffInformation::ypos (void) const [inline]

Get the row.

Returns:

the coeff's row

Definition at line 53 of file CoeffInformation.hh.

References m_yposition.

Referenced by WaveletTransform::getArea(), and WaveletTransform::getSubband().

8.12.3.3 int CoeffInformation::xpos (void) const [inline]

Get the col.

Returns:

the coeff's col

Definition at line 57 of file CoeffInformation.hh.

References m_xposition.

Referenced by WaveletTransform::getArea(), and WaveletTransform::getSubband().

8.12.3.4 int CoeffInformation::id (void) const [inline]

Get the ID.

Returns:

the id

Definition at line 61 of file CoeffInformation.hh.

References m_id.

8.12.3.5 void CoeffInformation::id (long *id*) [inline]

Set the ID.

Parameters:

id the id

Definition at line 64 of file CoeffInformation.hh.

References m_id.

8.12.3.6 void CoeffInformation::misc (coeff *misc*) [inline]

Set the misc data.

Parameters:

misc the misc data

Definition at line 67 of file CoeffInformation.hh.

References m_misc.

8.12.3.7 coeff CoeffInformation::misc (void) const [inline]

Get the misc data.

Returns:

the misc data

Definition at line 71 of file CoeffInformation.hh.

References m_misc.

8.12.3.8 int CoeffInformation::xypos (void) const [inline]

Get the absolute offset.

Returns:

the coeff's absolute offset

Definition at line 75 of file CoeffInformation.hh.

References m_xyposition.

8.12.3.9 void CoeffInformation::val (coeff *val*) [inline]

Assign a new value.

Parameters:

val the coeff's new value

Definition at line 79 of file CoeffInformation.hh.

References m_value.

8.12.3.10 void CoeffInformation::set (coeff *val*, int *y*, int *x*, int *abs*, long *id* = -1, coeff *misc* = 0.0)

Assign a new coefficient value only leaving the rest as is.

Parameters:

val the coeff's new value

y the coeff's row

x the coeff's col

abs the coeff's absolute position

id the new ID

misc the new misc data

8.12.3.11 void CoeffInformation::set (const CoeffInformation & *c*)

Copy a coefficient.

Parameters:

c the other coeff

8.12.3.12 void CoeffInformation::set (const Image & *img*, int *ypos*, int *xpos*)

Copy from an Image location.

Parameters:

img the image

ypos the row in the image

xpos the col in the image

8.12.3.13 int CoeffInformation::socmp (const CoeffInformation & *c*) const

Signed object comparison. If both values are identical, first the y, then the x component will be compared.

Parameters:

c the other coeff

Returns:

1: the other value is less, -1: the other value is greater, 2: the values are equal, but the other y component is less, -2: the values are equal, but the other y component is greater, 3: the values and y components are equal, but the other x component is less, -2: the values and y components are equal, but the other x component is greater, 0: both are equal

8.12.3.14 int CoeffInformation::aocmp (const CoeffInformation & *c*) const

Unsigned object comparison (absolute values). If both values are identical, first the y, then the x component will be compared.

Parameters:

c the other coeff

Returns:

1: the other value is less, -1: the other value is greater, 2: the values are equal, but the other y component is less, -2: the values are equal, but the other y component is greater, 3: the values and y components are equal, but the other x component is less, -2: the values and y components are equal, but the other x component is greater, 0: both are equal

8.12.3.15 int CoeffInformation::pcmp (const CoeffInformation & *c*) const

Position comparison.

Parameters:

c the other coeff

Returns:

2: the other y component is less, -2: the other y component is greater, 3: the other x component is less, -2: the other x component is greater, 0: both are equal

8.12.3.16 int CoeffInformation::svcmp (const CoeffInformation & *c*) const

Signed value comparison.

Parameters:

c the other coeff

Returns:

1: the other coeff is less, -1: the other coeff is greater, 0: both are equal

8.12.3.17 int CoeffInformation::avcmp (const CoeffInformation & *c*) const

Unsigned value comparison (absolute values).

Parameters:

c the other coeff

Returns:

1: the other coeff is less, -1: the other coeff is greater, 0: both are equal

8.12.3.18 bool CoeffInformation::equals (const CoeffInformation & *c*) const

Equality test. This includes both, the value and the coordinates.

Parameters:

c the other coeff

Returns:

both are equal: {*true}*, else {*false}*

Referenced by operator!=(), and operator==().

8.12.3.19 void CoeffInformation::dump (const char * *delim* = " ", const char * *file* = "") const

Writes all data to stdout or a file.

Exceptions:

ios_base::failure if the file could not be opened for writing.

Parameters:

delim what to print between two entries

file the name of the file (empty string for stdout)

8.12.3.20 bool CoeffInformation::operator== (const CoeffInformation & *c*) const [inline]

Convenience equality operator to avoid trouble in some cases.

Parameters:

c the other object

Returns:

true if the objects are equal

Definition at line 167 of file CoeffInformation.hh.

References equals().

8.12.3.21 bool CoeffInformation::operator!= (const CoeffInformation & *c*) const [inline]

Convenience inequality operator to avoid trouble in some cases.

Parameters:

c the other object

Returns:

true if the objects are not equal

Definition at line 172 of file CoeffInformation.hh.

References equals().

8.12.3.22 CoeffInformation& CoeffInformation::operator= (const CoeffInformation & *c*) [inline]

Convenience assignment operator to avoid trouble in some cases.

Parameters:

c the other object

Returns:

a reference to 'this'

Definition at line 177 of file CoeffInformation.hh.

8.12.4 Member Data Documentation

8.12.4.1 coeff CoeffInformation::m_value [private]

The coefficient value.

Definition at line 182 of file CoeffInformation.hh.

Referenced by val().

8.12.4.2 int CoeffInformation::m_yposition [private]

The coefficient row.

Definition at line 184 of file CoeffInformation.hh.

Referenced by ypos().

8.12.4.3 int CoeffInformation::m_xposition [private]

The coefficient col.

Definition at line 186 of file CoeffInformation.hh.

Referenced by xpos().

8.12.4.4 int CoeffInformation::m_xyposition [private]

The coefficient absolute offset.

Definition at line 188 of file CoeffInformation.hh.

Referenced by xypos().

8.12.4.5 long CoeffInformation::m_id [private]

The absolute ID (needed for tracking a coefficient.

Definition at line 190 of file CoeffInformation.hh.

Referenced by id().

8.12.4.6 coeff CoeffInformation::m_misc [private]

Other data for free use.

Definition at line 192 of file CoeffInformation.hh.

Referenced by misc().

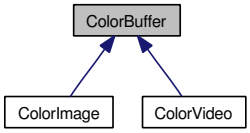
The documentation for this class was generated from the following file:

- CoeffInformation.hh

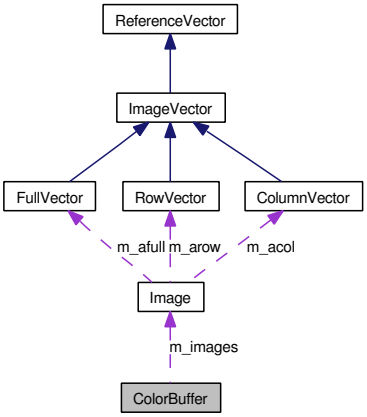
8.13 ColorBuffer Class Reference

`#include <ColorBuffer.hh>`

Inheritance diagram for ColorBuffer:



Collaboration diagram for ColorBuffer:



Public Member Functions

- ColorBuffer (int colors=3, clrmodel cmodel=cm_rgb)
- virtual ~ColorBuffer (void)
- virtual void swapColors (int ch1, int ch2)
- clrmodel colormodel (void) const
- int colors (void) const
- virtual void colormodel (clrmodel model)
- virtual coeff smax (void) const
- virtual coeff smin (void) const
- virtual int fileSize (char const *fname) const
- virtual bool equals (ColorBuffer &buf) const
- virtual bool epsilons (ColorBuffer &buf, coeff epsilon) const
- virtual void truncate (coeff min=0, coeff max=255)
- virtual void beautify (int yoffs=0, int xoffs=0, int ysize=0, int xsize=0)=0
- virtual coeff averageColor (int fromY=0, int fromX=0, int toY=-1, int toX=-1)
- virtual unsigned char * toPixmap (clrmodel cmodel)
- int rows (void) const

- int cols (void) const
- int size (void) const

Protected Member Functions

- void init (int colors, clrmodel cmodel=cm_rgb)
- virtual void destroy (void)
- virtual void valadjust (void)

Protected Attributes

- clrmodel m_cmodel
- int m_xsize
- int m_ysize
- int m_xysize
- int m_colors
- Image ** m_images
- bool m_isReference

8.13.1 Detailed Description

An abstract wrapper for color images and videos.

Definition at line 21 of file ColorBuffer.hh.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 ColorBuffer::ColorBuffer (int *colors* = 3, clrmodel *cmodel* = cm_rgb)

Constructor. Creates objects and sets actual start values.

Parameters:

colors the number of colors, and therefore the array's length

cmodel the color model

8.13.2.2 virtual ColorBuffer::~ColorBuffer (void) [virtual]

Destructor. Frees allocated objects.

8.13.3 Member Function Documentation

8.13.3.1 virtual void ColorBuffer::swapColors (int *ch1*, int *ch2*) [virtual]

Swap two color channels.

Exceptions:

invalid_argument if the channel specs are out of bounds

Parameters:

ch1 the first channel

ch2 the second channel

Reimplemented in ColorVideo.

8.13.3.2 clrmodel ColorBuffer::colormodel (void) const [inline]

Return the current colormodel

Exceptions:

invalid_argument number of colors is not 3, or unsupported format / conversion

Returns:

the colormodel

Definition at line 45 of file ColorBuffer.hh.

8.13.3.3 int ColorBuffer::colors (void) const [inline]

Return the current number of colors

Returns:

the number of colors

Definition at line 48 of file ColorBuffer.hh.

8.13.3.4 virtual void ColorBuffer::colormodel (clrmodel *model*) [virtual]

Convert the image to a new colormodel. Note: this operation is not completely invertable. Thus it should only be applied if no further transforms and no back-conversion are desired.

Parameters:

model the new colormodel ({*cm_rgb*} or {*cm_yuv*})

8.13.3.5 virtual coeff ColorBuffer::smax (void) const [virtual]

Returns the maximum value. Signs will be considered.

Exceptions:

invalid_argument empty image

Returns:

the maximum

Reimplemented in ColorVideo.

8.13.3.6 virtual coeff ColorBuffer::smin (void) const [virtual]

Returns the maximum value. Signs will be considered.

Exceptions:

invalid_argument empty image

Returns:

the maximum

Reimplemented in ColorVideo.

8.13.3.7 virtual int ColorBuffer::fileSize (char const * *fname*) const [virtual]

Returns the size in bytes of the file on the file system.

Parameters:

fname the file name

Returns:

the file size

8.13.3.8 virtual bool ColorBuffer::equals (ColorBuffer & *buf*) const [virtual]

Compares two images. Return {*true*} if both are equal.

Parameters:

buf The other {*ColorBuffer*} object

Returns:

if equals: {*true*}, else {*false*}

Reimplemented in ColorVideo.

8.13.3.9 virtual bool ColorBuffer::epsilons (ColorBuffer & *buf*, coeff *epsilon*) const [virtual]

Rough comparison. See if two images are similar according to a given {*epsilon*} (important for floating-point comparisons).

Parameters:

buf the other {*ColorBuffer*} object

epsilon the epsilon

Returns:

if both are identical: {*true*}, else {*false*}

Reimplemented in ColorVideo.

8.13.3.10 virtual void ColorBuffer::truncate (coeff *min* = 0, coeff *max* = 255) [virtual]

Do a dumb kind of beautification by just truncating values to their possible maxima depending on the image format or whatever. The original data will be overwritten, so this should only be used on clones or if the original data is not needed anymore.

Parameters:

min the minimum allowed value

max the maximum allowed value

Reimplemented in ColorVideo.

8.13.3.11 virtual void ColorBuffer::beautify (int *yoffs* = 0, int *xoffs* = 0, int *ysize* = 0, int *xsize* = 0) [pure virtual]

Do an intelligent kind of beautification considering previous transform steps. The original data will be overwritten, so this should only be used on clones or if the image is not going to be transformed afterwise. One region inside the image could be the low-pass subband and should thus be handled separately. If that is desired, the {*ysize*} and {*xsize*} parameters should be set to nonzero values.

Parameters:

ysize the horizontal size of the region that should be handled separately

xsize the vertical size of the region that should be handled separately

yoffs the horizontal offset of the region that should be handled separately

xoffs the vertical offset of the region that should be handled separately

Implemented in ColorImage, and ColorVideo.

8.13.3.12 virtual coeff ColorBuffer::averageColor (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) [virtual]

Return the average color for a rectangular region inside the image drawn from one point within and the second point just outside the region

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the average color

Reimplemented in ColorVideo.

8.13.3.13 virtual unsigned char* ColorBuffer::toPixmap (clrmodel *cmodel*) [virtual]

Export the image for other applications. This only works if the image has three channels!

Exceptions:

invalid_argument if the current colormodel or the one passed as parameter is unsupported.

Parameters:

cmodel the color model in which the image is exported. Currently {*cm_rgb*} and {*cm_yuv*} are supported.

Returns:

a new buffer containing the image pixels in the format according on the cmodel parameter

Reimplemented in ColorVideo.

8.13.3.14 int ColorBuffer::rows (void) const [inline]

Return the number of rows.

Returns:

the number of rows

Definition at line 144 of file ColorBuffer.hh.

8.13.3.15 int ColorBuffer::cols (void) const [inline]

Return the number of columns.

Returns:

the number of columns

Definition at line 148 of file ColorBuffer.hh.

8.13.3.16 int ColorBuffer::size (void) const [inline]

Return the overall size.

Returns:

the overall size

Definition at line 152 of file ColorBuffer.hh.

8.13.3.17 void ColorBuffer::init (int *colors*, clrmodel *cmodel* = cm_rgb) [protected]

Allocate objects and set standard values.

Parameters:

colors the number of colors

cmodel the initial color model

8.13.3.18 virtual void ColorBuffer::destroy (void) [protected, virtual]

Deletes allocated objects.

Reimplemented in ColorImage, and ColorVideo.

8.13.3.19 virtual void ColorBuffer::valadjust (void) [protected, virtual]

Adjust the coefficients to make them pixelizable.

8.13.4 Member Data Documentation

8.13.4.1 clrmodel ColorBuffer::m_cmodel [protected]

Definition at line 156 of file ColorBuffer.hh.

8.13.4.2 int ColorBuffer::m_xsize [protected]

The number of pixel cols

Definition at line 158 of file ColorBuffer.hh.

8.13.4.3 int ColorBuffer::m_ysize [protected]

The number of pixel rows

Definition at line 160 of file ColorBuffer.hh.

8.13.4.4 int ColorBuffer::m_xysize [protected]

The overall number of pixels

Definition at line 162 of file ColorBuffer.hh.

8.13.4.5 int ColorBuffer::m_colors [protected]

The number of colors

Definition at line 164 of file ColorBuffer.hh.

8.13.4.6 Image** ColorBuffer::m_images [protected]

An array of {*colors*} Image objects

Definition at line 166 of file ColorBuffer.hh.

Referenced by ColorVideo::currentFrameChannel().

8.13.4.7 bool ColorBuffer::m_isReference [protected]

True if the images are references and should not be deleted.

Definition at line 168 of file ColorBuffer.hh.

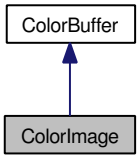
The documentation for this class was generated from the following file:

- ColorBuffer.hh

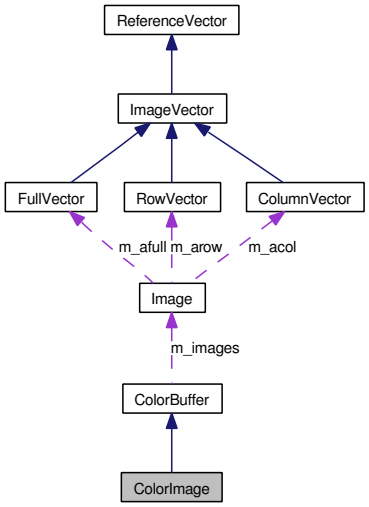
8.14 ColorImage Class Reference

`#include <ColorImage.hh>`

Inheritance diagram for ColorImage:



Collaboration diagram for ColorImage:



Public Member Functions

- ColorImage (int colors=3, clrmodel cmodel=cm_rgb, Image **images=NULL, bool isReference=false, bool isMine=false)
- ColorImage (int rows, int cols, int colors=3, clrmodel cmodel=cm_rgb)
- virtual ~ColorImage (void)
- Image & channel (int num)
- const Image & channel (int num) const
- void read (char const *fname)
- void read (char const *fname, filetype ftype)
- void write (char const *fname, bool beautify=false)
- void write (char const *fname, filetype ftype, bool beautify=false)
- Image * averageImage (void)
- virtual void beautify (int yoffs=0, int xoffs=0, int ysize=0, int xsize=0)
- ColorImage * clone (void) const

- virtual void copy (ColorImage &img)
- int quality (void) const
- void quality (int quality)
- ColorImage * crop (int yoffs, int xoffs, int ysize, int xsize) const
- virtual void paste (int yoffs, int xoffs, ColorImage &img)
- virtual void shift (int yoffs, int xoffs)
- void fill (coeff *values, int fromY=0, int fromX=0, int toY=-1, int toX=-1)
- void fill (int *values, int fromY=0, int fromX=0, int toY=-1, int toX=-1)
- virtual ColorImage * fitInto (int rows, int cols, int *fill=NULL, int function=0) const
- virtual ColorImage * scale (double factor, int function=0) const
- virtual ColorImage * scale (double yFactor, double xFactor, int function=0) const

Protected Member Functions

- void init (Image **images=NULL, bool isReference=false, bool isMine=false)
- virtual void destroy (void)

Private Attributes

- int m_quality

8.14.1 Detailed Description

An image wrapper for bitmap images of three color components. Objects may import and export different bitmap formats, such as PPM. The color models can be changed (RGB or YUV).

Definition at line 24 of file ColorImage.hh.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 ColorImage::ColorImage (int *colors* = `3`, clrmodel *cmodel* = `cm_rgb`, Image ** *images* = `NULL`, bool *isReference* = `false`, bool *isMine* = `false`)

Constructor. Creates objects and sets actual start values.

Parameters:

colors the number of colors, and therefore the array's length

cmodel the color model

images an array of {*colors*} images to copy the channel data from

ownership if true, only references to the images will be copied, and they will not be deleted at object destruction, else the images will be cloned and destroyed in the end

isMine if true the incoming images will not be cloned, i.e. ownership is transferred to this color image

8.14.2.2 ColorImage::ColorImage (int *rows*, int *cols*, int *colors* = 3, clrmodel *cmodel* = cm_rgb)

Constructor. Creates an image of a given size.

Parameters:

rows the number of image rows to create

cols the number of image cols to create

colors the number of colors

cmodel the color model

8.14.2.3 virtual ColorImage::~ColorImage (void) [virtual]

Destructor. Frees allocated objects.

8.14.3 Member Function Documentation

8.14.3.1 Image& ColorImage::channel (int *num*) [inline]

Returns a reference to one of the color channels. {*Don't* change the dimensions of single channels unless you know what you're doing!}

Parameters:

num the number of the color channel

Returns:

the color channel as a reference to an Image object

Definition at line 64 of file ColorImage.hh.

8.14.3.2 const Image& ColorImage::channel (int *num*) const [inline]

Returns a reference to one of the color channels. {*Don't* change the dimensions of single channels unless you know what you're doing!}

Parameters:

num the number of the color channel

Returns:

the color channel as a reference to an Image object

Definition at line 71 of file ColorImage.hh.

8.14.3.3 void ColorImage::read (char const * *fname*)

Read the image. All steps independent of the file format will be performed, like testing for file readability etc.

Exceptions:

invalid_argument invalid file format

ios_base::failure a read error has occured [not supported by all libraries, so eventually {*invalid_argument}* instead]

8.14.3.4 void ColorImage::read (char const * *fname*, filetype *ftype*)

Read an image. Reads an image from a file using the specified file type. Currently PGM, RAW and PFI are supported.

Exceptions:

invalid_argument the file type is either not supported or could not be determined from the given file name

Parameters:

fname the file name, if {*NULL}*, then {*stdin}*

ftype the file type

8.14.3.5 void ColorImage::write (char const * *fname*, bool *beautify* = `false`)

Write the image. All steps independent of the file format will be performed, like testing for file writeability etc.

Exceptions:

ios_base::failure a read error has occured [not supported by all libraries, so eventually {*invalid_argument}* instead]

invalid_argument the source image has a different number of colors than the target format allows

Parameters:

fname the target file name

beautify beautify images that have not had more analysis than synthesis steps?

8.14.3.6 void ColorImage::write (char const * *fname*, filetype *ftype*, bool *beautify* = `false`)

Write an image (abstract). Writes an image to a file using the specified file type.

Parameters:

ftype the file type

fname the file name, if {*NULL*}, then {*stdout*}

beautify beautify images that have not had more analysis than synthesis steps?

8.14.3.7 Image* ColorImage::averageImage (void)

Return a new greyscale image containing the three color channels' averages

Returns:

the new image

8.14.3.8 virtual void ColorImage::beautify (int *yoffs* = `0`, int *xoffs* = `0`, int *ysize* = `0`, int *xsize* = `0`) `[virtual]`

Do an intelligent kind of beautification considering previous transform steps. The original data will be overwritten, so this should only be used on clones or if the image is not going to be transformed afterwise. One region inside the image could be the low-pass subband and should thus be handled separately. If that is desired, the {*ysize*} and {*xsize*} parameters should be set to nonzero values.

Parameters:

ysize the horizontal size of the region that should be handled separately

xsize the vertical size of the region that should be handled separately

yoffs the horizontal offset of the region that should be handled separately

xoffs the vertical offset of the region that should be handled separately

Implements ColorBuffer.

8.14.3.9 ColorImage* ColorImage::clone (void) const

Produce a copy of this image.

Returns:

the copy

8.14.3.10 virtual void ColorImage::copy (ColorImage & *img*) [virtual]

Copy all data from a source image.

Parameters:

img the source image

Exceptions:

invalid_argument if the two images' number of color channels do not match

8.14.3.11 int ColorImage::quality (void) const [inline]

Return the current image quality factor.

Returns:

the current image quality factor

Definition at line 152 of file ColorImage.hh.

8.14.3.12 void ColorImage::quality (int *quality*) [inline]

Sets a new image quality factor.

Returns:

the new image quality factor

Definition at line 156 of file ColorImage.hh.

8.14.3.13 ColorImage* ColorImage::crop (int *yoffs*, int *xoffs*, int *ysize*, int *xsize*) const

Returns a subimage.

Parameters:

yoffs the row where to start

xoffs the col where to start

ysize the vertical size

xsize the horizontal size

Returns:

the new image

8.14.3.14 virtual void ColorImage::paste (int *yoffs*, int *xoffs*, ColorImage & *img*) [virtual]

Inserts a subimage.

Parameters:

yoffs the row where to start

xoffs the col where to start

img the image to insert

Exceptions:

invalid_argument if the two images' number of color channels do not match

8.14.3.15 virtual void ColorImage::shift (int *yoffs*, int *xoffs*) [virtual]

Shift an Image.

Parameters:

yoffs the rows to shift

xoffs the cols to shift

8.14.3.16 void ColorImage::fill (coeff * *values*, int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1)

Fills an image (or regions of it) with a given value

Parameters:

values an array of this->colors() entries containing the the new values one per channel for the selected positions (coeff values)

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

8.14.3.17 void ColorImage::fill (int * *values*, int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1)

Fills an image (or regions of it) with a given value

Parameters:

values an array of this->colors() entries containing the the new values one per channel for the selected positions (int values)

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

8.14.3.18 virtual ColorImage* ColorImage::fitInto (int *rows*, int *cols*, int * *fill* = NULL, int *function* = 0) const [virtual]

Produce a scaled version of the image which just fits into the given dimensions. If the aspect ratio does not fit into the new dimensions the remaining areas can be filled with a given color or alternatively the resulting image will be smaller than the target dimensions keeping its aspect ratio.

Exceptions:

invalid_argument the factor is negative or the function argument is invalid

Parameters:

rows the target number of rows

cols the target number of columns

fill if not NULL it must point to an array of as many values as color channels, so that each of the image's channels has its own fill greyscale value for the remaining space (else the a smaller image size will be chosen if the aspect ratio does not match)

function interpolation function (0: bilinear interpolation, 1: average, 2: nearest neighbour).

Returns:

a new rescaled image

8.14.3.19 virtual ColorImage* ColorImage::scale (double *factor*, int *function* = 0) const [virtual]

Produce a scaled version of the image. The aspect ratio will remain the same.

Exceptions:

invalid_argument the factor is negative or the function argument is invalid

Parameters:

factor the scale factor (>= 0, 1 for no change)

function interpolation function (0: bilinear interpolation, 1: average, 2: nearest neighbour).

Returns:

a new rescaled image

8.14.3.20 virtual ColorImage* ColorImage::scale (double *yFactor*, double *xFactor*, int *function* = 0) const [virtual]

Produce a scaled version of the image. The aspect ratio depends on the two scale factors

Exceptions:

invalid_argument the factor is negative or the function argument is invalid

Parameters:

yFactor the vertical scale factor (>= 0, 1 for no change)

xFactor the horizontal scale factor (>= 0, 1 for no change)

function interpolation function (0: bilinear interpolation, 1: average, 2: nearest neighbour).

Returns:

a new rescaled image

8.14.3.21 void ColorImage::init (Image ** *images* = NULL, bool *isReference* = false, bool *isMine* = false) [protected]

Allocate objects and set standard values.

Parameters:

images an array of {*colors*} images to copy the channel data from

isReference if true, only references to the images will be copied, and they will not be deleted at object destruction

isMine if true, the ownership of the incoming images will be transferred to this color image

8.14.3.22 virtual void ColorImage::destroy (void) [protected, virtual]

Deletes allocated objects.

Reimplemented from ColorBuffer.

8.14.4 Member Data Documentation

8.14.4.1 int ColorImage::m_quality [private]

The image quality (if stored in lossy format like JPG)

Definition at line 253 of file ColorImage.hh.

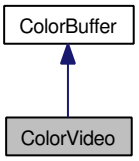
The documentation for this class was generated from the following file:

- ColorImage.hh

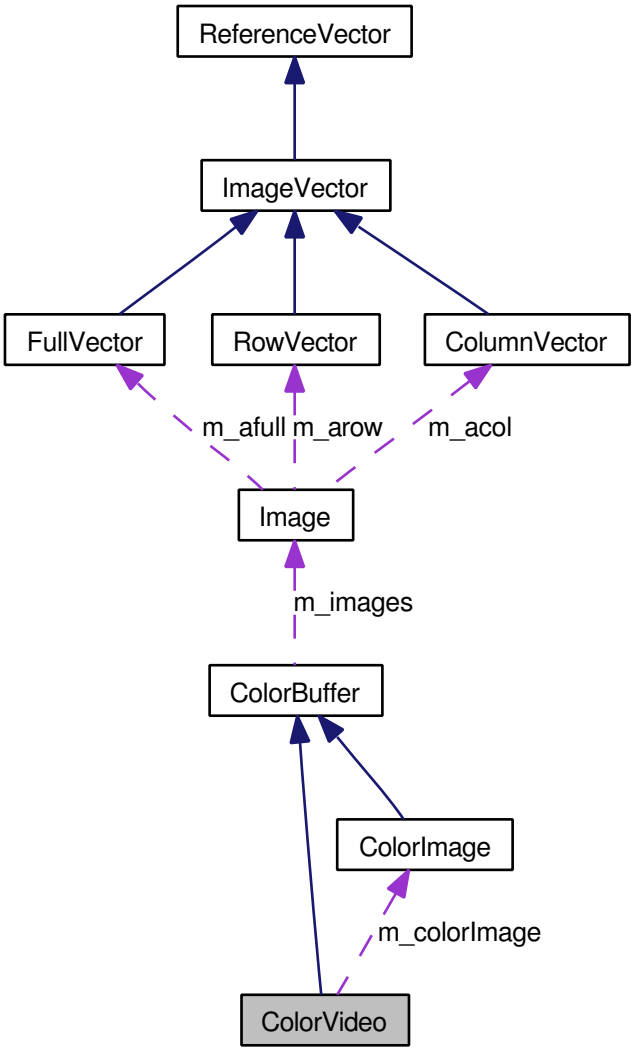
8.15 ColorVideo Class Reference

`#include <ColorVideo.hh>`

Inheritance diagram for ColorVideo:



Collaboration diagram for ColorVideo:



Public Member Functions

- ColorVideo (void)
- ColorVideo (const ColorVideo &cv)
- ColorVideo (int colors, clrmodel colormodel, int rows, int cols, int frames)
- virtual ~ColorVideo (void)
- virtual void swapColors (int ch1, int ch2)
- void colormodelVideo (clrmodel model)
- Image & currentFrameChannel (int num=0)
- ColorImage & currentFrame (void)
- double frameRate (void) const
- void frameRate (double fr)
- int current (void) const
- void current (int current)
- int frames (void) const
- void reframe (int frames, bool keephead)
- void resize (int rows, int cols)
- void read (char const *fname, int from=0, int to=0, int rawy=0, int rawx=0, int colors=3, clrmodel cmodel=cm_rgb, int skip=512)
- void read (char const *fname, filetype ftype, int from=0, int to=0, int rawy=0, int rawx=0, int colors=3, clrmodel cmodel=cm_rgb, int skip=512)
- void write (char const *fname, int rawskip=512, bool beautify=false)
- void write (char const *fname, filetype ftype, int rawskip=512, bool beautify=false)
- virtual coeff smax (void) const
- virtual coeff smin (void) const
- virtual bool equals (ColorBuffer &buf) const
- virtual bool epsilons (ColorBuffer &buf, coeff epsilon) const
- bool epsilonsFrames (int f1, int f2, coeff epsilon) const
- bool equalsFrames (int f1, int f2) const
- virtual void truncate (coeff min=0, coeff max=255)
- virtual void beautify (int yoffs=0, int xoffs=0, int ysize=0, int xsize=0)
- virtual coeff averageColor (int fromY=0, int fromX=0, int toY=-1, int toX=-1)
- virtual unsigned char * toPixmap (clrmodel cmodel)
- ColorVideo * clone (void) const

Static Public Member Functions

- static void videoDimensions (const char *fname, int &y, int &x, int &z, int colors, int skip=0)

Protected Member Functions

- virtual void destroy (void)
- bool epsilonsVideo (ColorVideo &cv, coeff epsilon) const
- void init (int colors, clrmodel cmodel, int rows, int cols, int frames)

Protected Attributes

- int m_frames
- int m_current
- int m_xyzsize
- VideoArray< coeff > ** m_arrays
- ColorImage * m_colorImage
- double m_frameRate

8.15.1 Detailed Description

An wrapper for videos containing several color components. Objects may import and export different video formats, such as RAW. The color models can be changed (RGB or YUV).

Definition at line 24 of file ColorVideo.hh.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 ColorVideo::ColorVideo (void)

Default Constructor. Creates objects and sets actual start values.

8.15.2.2 ColorVideo::ColorVideo (const ColorVideo & *cv*)

Copy Constructor.

Parameters:

cv the other color video

8.15.2.3 ColorVideo::ColorVideo (int *colors*, clrmodel *colormodel*, int *rows*, int *cols*, int *frames*)

Constructor setting a particular size etc. Initialization not done.

Parameters:

colors the number of colors

colormodel the initial color model

rows the number of rows

cols the number of cols

frames the number of frames

8.15.2.4 virtual ColorVideo::~ColorVideo (void) [virtual]

Destructor. Frees allocated objects.

8.15.3 Member Function Documentation

8.15.3.1 virtual void ColorVideo::swapColors (int *ch1*, int *ch2*) [virtual]

Swap two color channels.

Exceptions:

invalid_argument if the channel specs are out of bounds

Parameters:

ch1 the first channel

ch2 the second channel

Reimplemented from ColorBuffer.

8.15.3.2 void ColorVideo::colormodelVideo (clrmodel *model*)

Convert the video to a new colormodel. This applies the 'ColorBuffercolormodel()' operation on the whole video instead of only the current frame.

Parameters:

model the new colormodel ({*cm_rgb*} or {*cm_yuv*})

8.15.3.3 Image& ColorVideo::currentFrameChannel (int *num* = 0) [inline]

Returns a reference to one of the color channels of the current frame. {*Don't* change the dimensions of single channels unless you know what you're doing!}

Parameters:

num the number of the color channel

Returns:

the frame as a reference to an VideoFrame object

Definition at line 66 of file ColorVideo.hh.

References ColorBuffer::m_images.

8.15.3.4 ColorImage& ColorVideo::currentFrame (void) [inline]

Returns a reference to a color video containing the color channels of the current frame.

Returns:

the frame as a reference to an ColorImage object

Definition at line 71 of file ColorVideo.hh.

References m_colorImage.

8.15.3.5 double ColorVideo::frameRate (void) const [inline]

Returns the frame rate

Returns:

the frameRate

Definition at line 75 of file ColorVideo.hh.

References m_frameRate.

8.15.3.6 void ColorVideo::frameRate (double *fr*) [inline]

Sets the frame rate

Parameters:

fr the frameRate

Definition at line 79 of file ColorVideo.hh.

References m_frameRate.

8.15.3.7 int ColorVideo::current (void) const [inline]

Returns the number of the current frame.

Returns:

the current frame (counted from 0)

Definition at line 83 of file ColorVideo.hh.

References m_current.

8.15.3.8 void ColorVideo::current (int *current*)

Returns the number of the current frame.

Returns:

the current frame (counted from 0)

8.15.3.9 int ColorVideo::frames (void) const [inline]

Returns the number of frames

Returns:

the number of frames

Definition at line 91 of file ColorVideo.hh.

References m_frames.

8.15.3.10 void ColorVideo::reframe (int *frames*, bool *keephead*)

Sets new number of frames. The old frames will be copied to the new ones as far as they fit in. The new number must be greater than zero.

Exceptions:

invalid_argument the new number of frames is either negative or zero

Parameters:

frames the new number of frames.

keephead if shrinking the video, frames will be deleted at the end.

8.15.3.11 void ColorVideo::resize (int *rows*, int *cols*)

Resize the video. The old values will be copied to the new dimension as far as they fit in. The new dimensions must all be greater than zero.

Exceptions:

invalid_argument one or both dimensions are either negative or zero

Parameters:

rows the new number of rows

cols the new number of cols

8.15.3.12 void ColorVideo::read (char const * *fname*, int *from* = 0, int *to* = 0, int *rawy* = 0, int *rawx* = 0, int *colors* = 3, clrmodel *cmodel* = cm_rgb, int *skip* = 512)

Read the video. All steps independent of the file format will be performed, like testing for file readability etc.

Exceptions:

invalid_argument invalid file format

ios_base::failure a read error has occured [not supported by all libraries, so eventually {*invalid_argument}* instead]

Parameters:

fname the file name to read from

rawy the number of rows (only needed for RAW format)

rawx the number of cols (only needed for RAW format)

colors the number of colors (if known, necessary for raw videos)

from the frame to start with (from 0)

to one greater than the last frame, if equal to 'from' then the whole video will be read, if 0, then the video will be read from 'from' to the end

cmodel the color model

skip the amount of bytes to skip as a header before contents

8.15.3.13 void ColorVideo::read (char const * *fname*, filetype *ftype*, int *from* = 0, int *to* = 0, int *rawy* = 0, int *rawx* = 0, int *colors* = 3, clrmodel *cmodel* = cm_rgb, int *skip* = 512)

Read an video. Reads an video from a file using the specified file type. Currently PGM, RAW and PFI are supported.

Exceptions:

invalid_argument the file type is either not supported or could not be determined from the given file name

Parameters:

fname the file name, if {*NULL}*, then {*stdin}*

ftype the file type

rawy the number of rows (only needed for RAW format)

rawx the number of cols (only needed for RAW format)

colors the number of colors (if known, necessary for raw videos)

from the frame to start with (from 0)

to one greater than the last frame, if equal to 'from' then the whole video will be read, if 0, then the video will be read from 'from' to the end

cmodel the color model

skip the amount of bytes to skip as a header before contents

8.15.3.14 void ColorVideo::write (char const * *fname*, int *rawskip* = 512, bool *beautify* = false)

Write the video. All steps independent of the file format will be performed, like testing for file writeability etc.

Exceptions:

ios_base::failure a read error has occured [not supported by all libraries, so eventually {*invalid_argument}* instead]

invalid_argument the source video has a different number of colors than the target format allows

Parameters:

fname the target file name

rawskip the amount of header area before contents (will be filled with zeroes)

beautify beautify video that have not had more analysis than synthesis steps?

8.15.3.15 void ColorVideo::write (char const * *fname*, filetype *ftype*, int *rawskip* = 512, bool *beautify* = false)

Write an video. Writes an video to a file using the specified file type.

Parameters:

ftype the file type

fname the file name, if {*NULL}*, then {*stdout}*

rawskip the amount of header area before contents (will be filled with zeroes)

beautify beautify videos that have not had more analysis than synthesis steps?

8.15.3.16 virtual coeff ColorVideo::smax (void) const [virtual]

Returns the maximum value. Signs will be considered.

Exceptions:

invalid_argument empty video

Returns:

the maximum

Reimplemented from ColorBuffer.

8.15.3.17 virtual coeff ColorVideo::smin (void) const [virtual]

Returns the maximum value. Signs will be considered.

Exceptions:

invalid_argument empty video

Returns:

the maximum

Reimplemented from ColorBuffer.

8.15.3.18 virtual bool ColorVideo::equals (ColorBuffer & *buf*) const [virtual]

Compares two videos. Return {*true*} if both are equal.

Parameters:

buf The other {*ColorBuffer*} object

Returns:

if equals: {*true*}, else {*false*}

Reimplemented from ColorBuffer.

8.15.3.19 virtual bool ColorVideo::epsilons (ColorBuffer & *buf*, coeff *epsilon*) const [virtual]

Rough comparison. See if two videos are similar according to a given {*epsilon*} (important for floating-point comparisons).

Parameters:

buf the other {*ColorBuffer*} object

epsilon the epsilon

Returns:

if both are identical: {*true*}, else {*false*}

Reimplemented from ColorBuffer.

8.15.3.20 bool ColorVideo::epsilonsFrames (int *f1*, int *f2*, coeff *epsilon*) const

Rough comparison. See if this and another frame are similar according to a given {*epsilon*} (important for floating-point comparisons).

Exceptions:

invalid_argument one of the frames is out of bounds

Parameters:

f1 the first frame

f2 the second frame

epsilon the epsilon

Returns:

if both are identical: {*true*}, else {*false*}

8.15.3.21 bool ColorVideo::equalsFrames (int *f1*, int *f2*) const

Exact comparison. See if two frames are similar.

Exceptions:

invalid_argument one of the frames is out of bounds

Parameters:

f1 the first frame

f2 the second frame if both are identical: {*true*}, else {*false*}

8.15.3.22 virtual void ColorVideo::truncate (coeff *min* = `0`, coeff *max* = `255`) `[virtual]`

Do a dumb kind of beautification by just truncating values to their possible maxima depending on the video format or whatever. The original data will be overwritten, so this should only be used on clones or if the original data is not needed anymore.

Parameters:

min the minimum allowed value

max the maximum allowed value

Reimplemented from ColorBuffer.

8.15.3.23 virtual void ColorVideo::beautify (int *yoffs* = `0`, int *xoffs* = `0`, int *ysize* = `0`, int *xsize* = `0`) `[virtual]`

Do an intelligent kind of beautification considering previous transform steps. The original data will be overwritten, so this should only be used on clones or if the video is not going to be transformed afterwise. One region inside the video could be the low-pass subband and should thus be handled separately. If that is desired, the {*ysize*} and {*xsize*} parameters should be set to nonzero values. Note that the current implementation is very memory consuming.

Parameters:

ysize the horizontal size of the region that should be handled separately

xsize the vertical size of the region that should be handled separately

yoffs the horizontal offset of the region that should be handled separately

xoffs the vertical offset of the region that should be handled separately

Implements ColorBuffer.

8.15.3.24 virtual coeff ColorVideo::averageColor (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) [virtual]

Return the average color for a rectangular region inside the video drawn from one point within and the second point just outside the region

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower video border)

toX the second point's col (-1 if right video border)

Returns:

the average color

Reimplemented from ColorBuffer.

8.15.3.25 virtual unsigned char* ColorVideo::toPixmap (clrmodel *cmodel*) [virtual]

Export the video for other applications. This only works if the video has three channels!

Exceptions:

invalid_argument if the current colormodel or the one passed as parameter is unsupported.

Parameters:

cmodel the color model in which the video is exported. Currently {*cm_rgb*} and {*cm_yuv*} are supported.

Returns:

a new buffer containing the video pixels in the format according on the cmodel parameter

Reimplemented from ColorBuffer.

8.15.3.26 ColorVideo* ColorVideo::clone (void) const

Produce a copy of this video

Returns:

the copy

8.15.3.27 static void ColorVideo::videoDimensions (const char * *fname*, int & *y*, int & *x*, int & *z*, int *colors*, int *skip* = 0) [static]

Try to determine some of a video file's dimensions: If the file is a raw video, two out of the three need to be specified, for AVI files all three can be determined from the header.

Exceptions:

invalid_argument if there was an error

Parameters:

fname the file name of the video

y the video's rows

x the video's columns

z the video's number of frames

colors the number of colors (always 3 with AVI files)

skip the size of the header (only raw videos)

8.15.3.28 virtual void ColorVideo::destroy (void) [protected, virtual]

Deletes allocated objects.

Reimplemented from ColorBuffer.

8.15.3.29 bool ColorVideo::epsilonsVideo (ColorVideo & *cv*, coeff *epsilon*) const [protected]

Compares two videos

Parameters:

cv the other video

epsilon the min difference

Returns:

true if they equal according to epsilon

8.15.3.30 void ColorVideo::init (int *colors*, clrmodel *cmodel*, int *rows*, int *cols*, int *frames*) [protected]

Allocate objects and set standard values.

Exceptions:

invalid_argument if any of the arguments does not make sense

Parameters:

colors the number of colors

cmodel the initial color model

rows the number of rows

cols the number of cols

frames the number of frames

8.15.4 Member Data Documentation

8.15.4.1 int ColorVideo::m_frames [protected]

A the number of frames

Definition at line 322 of file ColorVideo.hh.

Referenced by frames().

8.15.4.2 int ColorVideo::m_current [protected]

A counter for the current frame

Definition at line 324 of file ColorVideo.hh.

Referenced by current().

8.15.4.3 int ColorVideo::m_xyzsize [protected]

The video's total size.

Definition at line 326 of file ColorVideo.hh.

8.15.4.4 VideoArray< coeff >** ColorVideo::m_arrays [protected]

An array of {*colors}* Image objects

Definition at line 328 of file ColorVideo.hh.

8.15.4.5 ColorImage* ColorVideo::m_colorImage [protected]

All channels of the current frame bundled in a ColorImage object.

Definition at line 330 of file ColorVideo.hh.

Referenced by currentFrame().

8.15.4.6 double ColorVideo::m_frameRate [protected]

The frame rate (if known)

Definition at line 332 of file ColorVideo.hh.

Referenced by frameRate().

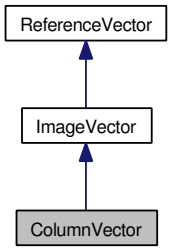
The documentation for this class was generated from the following file:

- ColorVideo.hh

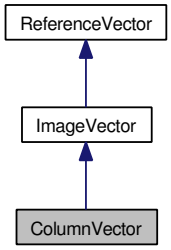
8.16 ColumnVector Class Reference

`#include <ColumnVector.hh>`

Inheritance diagram for ColumnVector:



Collaboration diagram for ColumnVector:



Public Member Functions

- ColumnVector (ImageArray< coeff > *ar)
- virtual ~ColumnVector (void)
- virtual void go (int root)
- virtual coeff at (int pos)
- virtual void to (int pos, coeff val)
- virtual int size (void)
- virtual void update (void)

8.16.1 Detailed Description

Column-reference. A reference to a two-dimensional array's column.

Definition at line 21 of file ColumnVector.hh.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 ColumnVector::ColumnVector (ImageArray< coeff > * *ar*) [inline]

Constructor. As we've got no default constructor we need to call the mother's class' constructor here...

Definition at line 28 of file ColumnVector.hh.

References go().

8.16.2.2 virtual ColumnVector::~ColumnVector (void) [inline, virtual]

Destructor. Does nothing.

Definition at line 30 of file ColumnVector.hh.

8.16.3 Member Function Documentation

8.16.3.1 virtual void ColumnVector::go (int *root*) [virtual]

Sets a new root. In this context this "new root" means a new col.

Exceptions:

invalid_argument the new root is greater than the number of columns

Parameters:

root the new row

Reimplemented from ImageVector.

Referenced by ColumnVector().

8.16.3.2 virtual coeff ColumnVector::at (int *pos*) [virtual]

Get a value. Return the value at a given column from the current row.

Exceptions:

invalid_argument a negative value was given for the new root

Parameters:

pos the column

Returns:

the value

Implements ImageVector.

8.16.3.3 virtual void ColumnVector::to (int *pos*, coeff *val*) [virtual]

Assigns a new value. Sets a new value at a given column from the current column.

Parameters:

pos the column

val the new value

Implements ImageVector.

8.16.3.4 virtual int ColumnVector::size (void) [virtual]

The vector's size. Returns the vector's size (the number of cols)

Returns:

the number of cols

Implements ImageVector.

8.16.3.5 virtual void ColumnVector::update (void) [virtual]

Updates the vector's settings. This is necessary each time after the array has been resized.

Reimplemented from ImageVector.

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

- ColumnVector.hh

8.17 common_struct Struct Reference

`#include <avilib.h>`

Public Attributes

- uint16_t wFormatTag
- uint16_t wChannels
- uint32_t dwSamplesPerSec
- uint32_t dwAvgBytesPerSec
- uint16_t wBlockAlign
- uint16_t wBitsPerSample

8.17.1 Detailed Description

Definition at line 485 of file avilib.h.

8.17.2 Member Data Documentation

8.17.2.1 uint16_t common_struct::wFormatTag

Definition at line 487 of file avilib.h.

8.17.2.2 uint16_t common_struct::wChannels

Definition at line 488 of file avilib.h.

8.17.2.3 uint32_t common_struct::dwSamplesPerSec

Definition at line 489 of file avilib.h.

8.17.2.4 uint32_t common_struct::dwAvgBytesPerSec

Definition at line 490 of file avilib.h.

8.17.2.5 uint16_t common_struct::wBlockAlign

Definition at line 491 of file avilib.h.

8.17.2.6 uint16_t common_struct::wBitsPerSample

Definition at line 492 of file avilib.h.

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

- avilib.h

8.18 FileName Class Reference

`#include <FileName.hh>`

Public Member Functions

- FileName (char const *name)
- ~FileName (void)
- bool isext (const char *ext) const
- void toext (const char *ext)
- filetype guess (void) const
- char * file (void) const
- char * dir (void) const
- char * base (void) const
- char * ext (void) const
- bool dos (void) const

Private Member Functions

- void guessext (void)

Private Attributes

- char * m_fname
- char * m_dirname
- char * m_basename
- char * m_extension
- bool m_isdos
- filetype m_ftype

8.18.1 Detailed Description

File names. This splits a DOS- or Unix-like filespec into its parts (dir, name without extension, extension) and guesses what kind of (image) file it is according to the (case insignificant) file extension.

Definition at line 30 of file FileName.hh.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 FileName::FileName (char const * *name*)

Constructor. The file name is being set, and the filename is split into its bits. At last the file type is guessed.

Parameters:

name the file name

Exceptions:

invalid_argument the file name is a {*NULL*} pointer

8.18.2.2 FileName::~FileName (void)

Destructor. Releases allocated memory.

8.18.3 Member Function Documentation

8.18.3.1 bool FileName::isext (const char * *ext*) const

Compares two extensions. The comparison is case insensitive.

Parameters:

ext the other extension

Returns:

they are equal: {*true*} else {*false*}

8.18.3.2 void FileName::toext (const char * *ext*)

Set a new extension. This might be useful if we want to create an output file name.

Parameters:

ext the new extension

8.18.3.3 filetype FileName::guess (void) const [inline]

Return the guessed type. No calculation is done here.

Returns:

the file type

Definition at line 59 of file FileName.hh.

References m_ftype.

8.18.3.4 char* FileName::file (void) const [inline]

Return the file name.

Returns:

the file name

Definition at line 63 of file FileName.hh.

References m_fname.

8.18.3.5 char* FileName::dir (void) const [inline]

Return the directory name.

Returns:

the directory name

Definition at line 67 of file FileName.hh.

References m_dirname.

8.18.3.6 char* FileName::base (void) const [inline]

Return the basename without extension.

Returns:

the name without extension

Definition at line 71 of file FileName.hh.

References m_basename.

8.18.3.7 char* FileName::ext (void) const [inline]

Return the file extension.

Returns:

the file extension

Definition at line 75 of file FileName.hh.

References m_extension.

8.18.3.8 bool FileName::dos (void) const [inline]

Is it a DOS-like name?

Returns:

if a DOS-like name: {*true}*, else {*false}*

Definition at line 79 of file FileName.hh.

References m_isdos.

8.18.3.9 void FileName::guessext (void) [private]

Guess the type. This sets the {*m_ftype}* variable.

8.18.4 Member Data Documentation

8.18.4.1 char* FileName::m_fname [private]

The complete filename

Definition at line 83 of file FileName.hh.

Referenced by file().

8.18.4.2 char* FileName::m_dirname [private]

The directory name

Definition at line 85 of file FileName.hh.

Referenced by dir().

8.18.4.3 char* FileName::m_basename [private]

The file name without extension

Definition at line 87 of file FileName.hh.

Referenced by base().

8.18.4.4 char* FileName::m_extension [private]

The file extension

Definition at line 89 of file FileName.hh.

Referenced by ext().

8.18.4.5 bool FileName::m_isdos [private]

Is it a DOS-like filespec? It is as soon as we find a backslash somewhere in it!

Definition at line 92 of file FileName.hh.

Referenced by dos().

8.18.4.6 filetype FileName::m_ftype [private]

The file type. This is guessed from the file's extension.

Definition at line 94 of file FileName.hh.

Referenced by guess().

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

- FileName.hh

8.19 Filter Class Reference

`#include <Filter.hh>`

Public Member Functions

- Filter (void)
- Filter (int m_size, int firstIndex=0, coeff *coeffs=NULL)
- Filter (const Filter &filter)
- ~Filter (void)
- void init (int m_size, int filterFirst, coeff *coeffs)
- void dump (void)
- coeff at (int index)
- int fsize (void)
- int first (void)

Protected Member Functions

- void copy (const Filter &filter)
- void tof (int index, coeff value)
- coeff atf (int index)

Protected Attributes

- int m_size
- int m_firstIndex
- coeff * m_coeffs

Friends

- class FilterSet

8.19.1 Detailed Description

Definition at line 42 of file Filter.hh.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 Filter::Filter (void) [inline]

Definition at line 47 of file Filter.hh.

References m_coeffs, m_firstIndex, m_size, and NULL.

8.19.2.2 Filter::Filter (int *m_size*, int *firstIndex* = 0, coeff * *coeffs* = NULL) [inline]

Definition at line 48 of file Filter.hh.

References init().

8.19.2.3 Filter::Filter (const Filter & *filter*) [inline]

Definition at line 50 of file Filter.hh.

References copy(), m_coeffs, and NULL.

8.19.2.4 Filter::~Filter (void)

8.19.3 Member Function Documentation

8.19.3.1 void Filter::init (int *m_size*, int *filterFirst*, coeff * *coeffs*)

Referenced by Filter().

8.19.3.2 void Filter::dump (void)

8.19.3.3 coeff Filter::at (int *index*) [inline]

Definition at line 55 of file Filter.hh.

References m_coeffs.

8.19.3.4 int Filter::fsize (void) [inline]

Definition at line 56 of file Filter.hh.

References m_size.

8.19.3.5 int Filter::first (void) [inline]

Definition at line 57 of file Filter.hh.

References m_firstIndex.

8.19.3.6 void Filter::copy (const Filter & *filter*) [protected]

Referenced by Filter().

8.19.3.7 void Filter::tof (int *index*, coeff *value*) [inline, protected]

Definition at line 64 of file Filter.hh.

References m_coeffs, and m_firstIndex.

8.19.3.8 coeff Filter::atf (int *index*) [inline, protected]

Definition at line 66 of file Filter.hh.

References m_coeffs, and m_firstIndex.

8.19.4 Friends And Related Function Documentation

8.19.4.1 friend class FilterSet [friend]

Definition at line 44 of file Filter.hh.

8.19.5 Member Data Documentation

8.19.5.1 int Filter::m_size [protected]

Definition at line 60 of file Filter.hh.

Referenced by Filter(), and fsize().

8.19.5.2 int Filter::m_firstIndex [protected]

Definition at line 61 of file Filter.hh.

Referenced by atf(), Filter(), first(), and tof().

8.19.5.3 coeff* Filter::m_coeffs [protected]

Definition at line 63 of file Filter.hh.

Referenced by at(), atf(), Filter(), and tof().

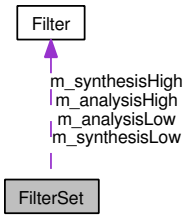
The documentation for this class was generated from the following file:

- Filter.hh

8.20 FilterSet Class Reference

`#include <Filter.hh>`

Collaboration diagram for FilterSet:



Public Member Functions

- FilterSet (void)
- FilterSet (bool symmetric, coeff *anLow, int anLowSize, int anLowFirst, coeff *synLow=NULL, int synLowSize=0, int synLowFirst=0)
- FilterSet (const FilterSet &filterset)
- ~FilterSet (void)
- Filter & alow (void)
- Filter & ahigh (void)
- Filter & slow (void)
- Filter & shigh (void)
- bool issym (void)
- void dump (void)
- void init (bool symmetric, coeff *anLow, int anLowSize, int anLowFirst, coeff *synLow, int synLowSize, int synLowFirst)

Static Public Member Functions

- static FilterSet & filterFromString (char *str)
- static const char * filterToString (FilterSet &filter)

Protected Member Functions

- void copy (const FilterSet &filterset)

Protected Attributes

- bool m_symmetric
- Filter * m_analysisLow
- Filter * m_analysisHigh
- Filter * m_synthesisLow
- Filter * m_synthesisHigh

8.20.1 Detailed Description

Originally this was wavelet.hh from Geoff Davis' Wavelet construction kit. This version has been adapted to my classes resulting in small changes to names and formatting etc. I also removed the class Wavelet which looks different in my code!

Definition at line 79 of file Filter.hh.

8.20.2 Constructor & Destructor Documentation

8.20.2.1 FilterSet::FilterSet (void) [inline]

Definition at line 82 of file Filter.hh.

References m_analysisHigh, m_analysisLow, m_symmetric, m_synthesisHigh, m_synthesisLow, and NULL.

8.20.2.2 FilterSet::FilterSet (bool *symmetric*, coeff * *anLow*, int *anLowSize*, int *anLowFirst*, coeff * *synLow* = NULL, int *synLowSize* = 0, int *synLowFirst* = 0)

8.20.2.3 FilterSet::FilterSet (const FilterSet & *filterset*)

8.20.2.4 FilterSet::~FilterSet (void)

8.20.3 Member Function Documentation

8.20.3.1 Filter& FilterSet::alow (void) [inline]

Definition at line 91 of file Filter.hh.

References m_analysisLow.

8.20.3.2 Filter& FilterSet::ahigh (void) [inline]

Definition at line 92 of file Filter.hh.

References m_analysisHigh.

8.20.3.3 Filter& FilterSet::slow (void) [inline]

Definition at line 93 of file Filter.hh.

References m_synthesisLow.

8.20.3.4 Filter& FilterSet::shigh (void) [inline]

Definition at line 94 of file Filter.hh.

References m_synthesisHigh.

8.20.3.5 bool FilterSet::issym (void) [inline]

Definition at line 95 of file Filter.hh.

References m_symmetric.

8.20.3.6 void FilterSet::dump (void)

8.20.3.7 void FilterSet::init (bool *symmetric*, coeff * *anLow*, int *anLowSize*, int *anLowFirst*, coeff * *synLow*, int *synLowSize*, int *synLowFirst*)

8.20.3.8 static FilterSet& FilterSet::filterFromString (char * *str*) [static]

Return a FilterSet from a String.

Exceptions:

invalid_argument the filter in the string was not found

Returns:

a reference to the filter set

8.20.3.9 static const char* FilterSet::filterToString (FilterSet & *filter*) [static]

8.20.3.10 void FilterSet::copy (const FilterSet & *filterset*) [protected]

8.20.4 Member Data Documentation

8.20.4.1 bool FilterSet::m_symmetric [protected]

Definition at line 112 of file Filter.hh.

Referenced by FilterSet(), and issym().

8.20.4.2 Filter* FilterSet::m_analysisLow [protected]

Definition at line 113 of file Filter.hh.

Referenced by alow(), and FilterSet().

8.20.4.3 Filter* FilterSet::m_analysisHigh [protected]

Definition at line 114 of file Filter.hh.

Referenced by ahigh(), and FilterSet().

8.20.4.4 Filter* FilterSet::m_synthesisLow [protected]

Definition at line 115 of file Filter.hh.

Referenced by FilterSet(), and slow().

8.20.4.5 Filter* FilterSet::m_synthesisHigh [protected]

Definition at line 116 of file Filter.hh.

Referenced by FilterSet(), and shigh().

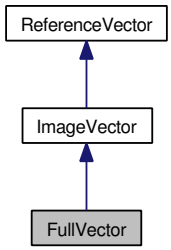
The documentation for this class was generated from the following file:

- Filter.hh

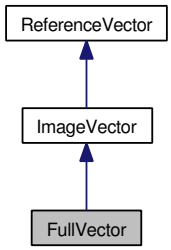
8.21 FullVector Class Reference

`#include <FullVector.hh>`

Inheritance diagram for FullVector:



Collaboration diagram for FullVector:



Public Member Functions

- FullVector (ImageArray< coeff > *ar)
- virtual ~FullVector (void)
- virtual void go (int root)
- virtual coeff at (int pos)
- virtual void to (int pos, coeff val)
- virtual int size (void)
- virtual void update (void)

Private Attributes

- int m_xysize

8.21.1 Detailed Description

Full reference. A reference to a one-dimensional array.

Definition at line 21 of file FullVector.hh.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 FullVector::FullVector (ImageArray< coeff > * *ar*) [inline]

Constructor. As we've got no default constructor we need to call the mother's class' constructor here...

Definition at line 28 of file FullVector.hh.

References go(), ImageVector::m_xsize, m_xysize, and ImageVector::m_ysize.

8.21.2.2 virtual FullVector::~FullVector (void) [inline, virtual]

Destructor. Does nothing.

Definition at line 31 of file FullVector.hh.

8.21.3 Member Function Documentation

8.21.3.1 virtual void FullVector::go (int *root*) [virtual]

Sets a new root. In this context this "new root" means a new row.

Exceptions:

invalid_argument the new root is greater than the number of rows

Parameters:

root the new row

Reimplemented from ImageVector.

Referenced by FullVector().

8.21.3.2 virtual coeff FullVector::at (int *pos*) [virtual]

Get a value. Return the value at a given column from the current row.

Exceptions:

invalid_argument a negative value was given for the new root

Parameters:

pos the column

Returns:

the value

Implements ImageVector.

8.21.3.3 virtual void FullVector::to (int *pos*, coeff *val*) [virtual]

Assigns a new value. Sets a new value at a given column from the current row.

Parameters:

pos the column

val the new value

Implements ImageVector.

8.21.3.4 virtual int FullVector::size (void) [virtual]

The vector's size. Returns the vector's size (the number of cols)

Returns:

the number of cols

Implements ImageVector.

8.21.3.5 virtual void FullVector::update (void) [virtual]

Updates the vector's settings. This is necessary each time after the array has been resized.

Reimplemented from ImageVector.

8.21.4 Member Data Documentation

8.21.4.1 int FullVector::m_xysize [private]

The encapsulated array's size.

Definition at line 62 of file FullVector.hh.

Referenced by FullVector().

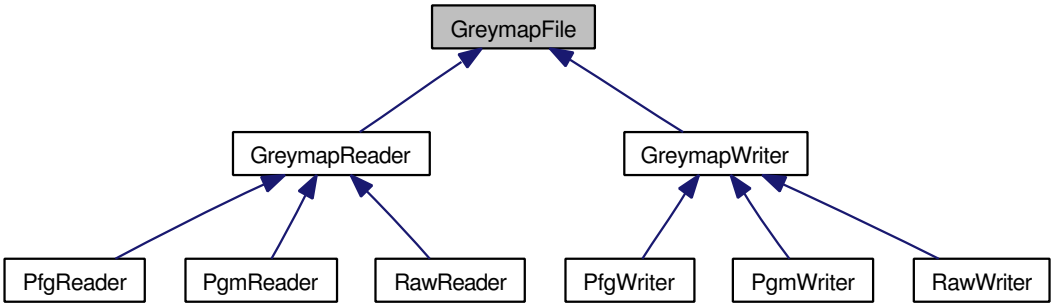
The documentation for this class was generated from the following file:

- FullVector.hh

8.22 GreymapFile Class Reference

`#include <GreymapFile.hh>`

Inheritance diagram for GreymapFile:



Public Member Functions

- GreymapFile (char const *name, ImageArray< coeff > &data)
- virtual ~GreymapFile (void)

Protected Attributes

- ImageArray< pixel > * m_pixels
- ImageArray< coeff > * m_coeffs
- char const * m_fname

8.22.1 Detailed Description

An abstract image file. A framework to create readers and writers on any greyscale images.

Definition at line 24 of file GreymapFile.hh.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 GreymapFile::GreymapFile (char const * *name*, ImageArray< coeff > & *data*)

Constructor. Initializes internal fields and gets an {*ImageArray*} object that may already contain an image or will get one later.

Parameters:

name the file name

data the array for the values

8.22.2.2 virtual GreymapFile::~GreymapFile (void) [virtual]

Destructor. Releases some memory.

8.22.3 Member Data Documentation

8.22.3.1 ImageArray<pixel>* GreymapFile::m_pixels [protected]

The pixels. These are one-byte integer values representing pixels stored in typical image formats.

Definition at line 44 of file GreymapFile.hh.

8.22.3.2 ImageArray<coeff>* GreymapFile::m_coeffs [protected]

Coefficients. These are long floating-point numbers to represent positive or negative floating-point values created through a transform.

Definition at line 48 of file GreymapFile.hh.

8.22.3.3 char const* GreymapFile::m_fname [protected]

The file name. The name of the file associated with this object.

Definition at line 50 of file GreymapFile.hh.

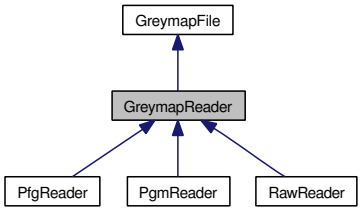
The documentation for this class was generated from the following file:

- GreymapFile.hh

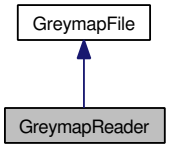
8.23 GreymapReader Class Reference

`#include <GreymapReader.hh>`

Inheritance diagram for GreymapReader:



Collaboration diagram for GreymapReader:



Public Member Functions

- GreymapReader (char const *name, ImageArray< coeff > &data)
- virtual ~GreymapReader (void)
- virtual void read (void)

Protected Member Functions

- virtual int readfmt (void)=0

Private Member Functions

- void unpixel (void)

8.23.1 Detailed Description

An abstract greyscale image reader. The image is being read putting its content into an {*ImageArray*} object. A reference to such an object is passed by the constructor, the actual object belongs to the outer world.

Definition at line 25 of file GreymapReader.hh.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 GreymapReader::GreymapReader (char const * *name*, ImageArray< coeff > & *data*) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference.

Parameters:

name the file name, if {*NULL*}, then {*stdin*}

data the reference to the {*ImageArray*} object

Definition at line 36 of file GreymapReader.hh.

8.23.2.2 virtual GreymapReader::~GreymapReader (void) [inline, virtual]

Destructor - does nothing.

Definition at line 40 of file GreymapReader.hh.

8.23.3 Member Function Documentation

8.23.3.1 virtual void GreymapReader::read (void) [virtual]

Read the image. All steps independent of the file format will be performed, like testing for file readability etc.

Exceptions:

invalid_argument invalid file format

ios_base::failure a read error has occured [not supported by all libraries, so eventually {*invalid_argument*} instead]

8.23.3.2 void GreymapReader::unpixel (void) [private]

Fill the coefficient array. If the file contains pixels rather than coefficients (like any bitmap format) this function will be called to create and fill the coefficient array.

8.23.3.3 virtual int GreymapReader::readfmt (void) [protected, pure virtual]

Read different file formats (abstract). This is the method to be implemented for every image file format. It will be called by the {*read* ()} method.

Returns:

0 if successful, -1 on read error, -2 on file format error.

Implemented in PfgReader, PgmReader, and RawReader.

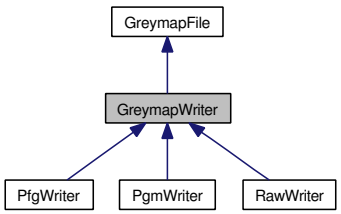
The documentation for this class was generated from the following file:

- GreymapReader.hh

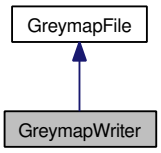
8.24 GreymapWriter Class Reference

`#include <GreymapWriter.hh>`

Inheritance diagram for GreymapWriter:



Collaboration diagram for GreymapWriter:



Public Member Functions

- GreymapWriter (char const *name, ImageArray< coeff > &data)
- virtual ~GreymapWriter (void)
- void write (void)

Protected Member Functions

- virtual int writefmt (void)=0

Private Member Functions

- void pixelize (void)

8.24.1 Detailed Description

An abstract image writer. The image is being written putting its content from an {*ImageArray*} object into the file. A reference to the object holding the image is passed by the constructor, the actual object belongs to the outer world.

Definition at line 25 of file GreymapWriter.hh.

8.24.2 Constructor & Destructor Documentation

8.24.2.1 GreymapWriter::GreymapWriter (char const * *name*, ImageArray< coeff > & *data*) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference.

Parameters:

name the file name, if {*NULL*}, then {*stdout*}

data the reference to the {*ImageArray*} object

Definition at line 36 of file GreymapWriter.hh.

8.24.2.2 virtual GreymapWriter::~GreymapWriter (void) [inline, virtual]

Destructor - does nothing.

Definition at line 40 of file GreymapWriter.hh.

8.24.3 Member Function Documentation

8.24.3.1 void GreymapWriter::write (void)

Write the image. All steps independent of the file format will be performed, like testing for file writeability etc.

Exceptions:

ios_base::failure a read error has occured [not supported by all libraries, so eventually {*invalid_argument*} instead]

8.24.3.2 void GreymapWriter::pixelize (void) [private]

Fill the pixels array. This is not trivial as the coefficients may contain negative values that have to be represented as grey-scale pixels.

8.24.3.3 virtual int GreymapWriter::writefmt (void) [protected, pure virtual]

Write different file formats (abstract). This is the method to be implemented for every image file format. It will be called by the {*write* ()} method.

Returns:

0 if successful, -1 on write error.

Implemented in PfgWriter, PgmWriter, and RawWriter.

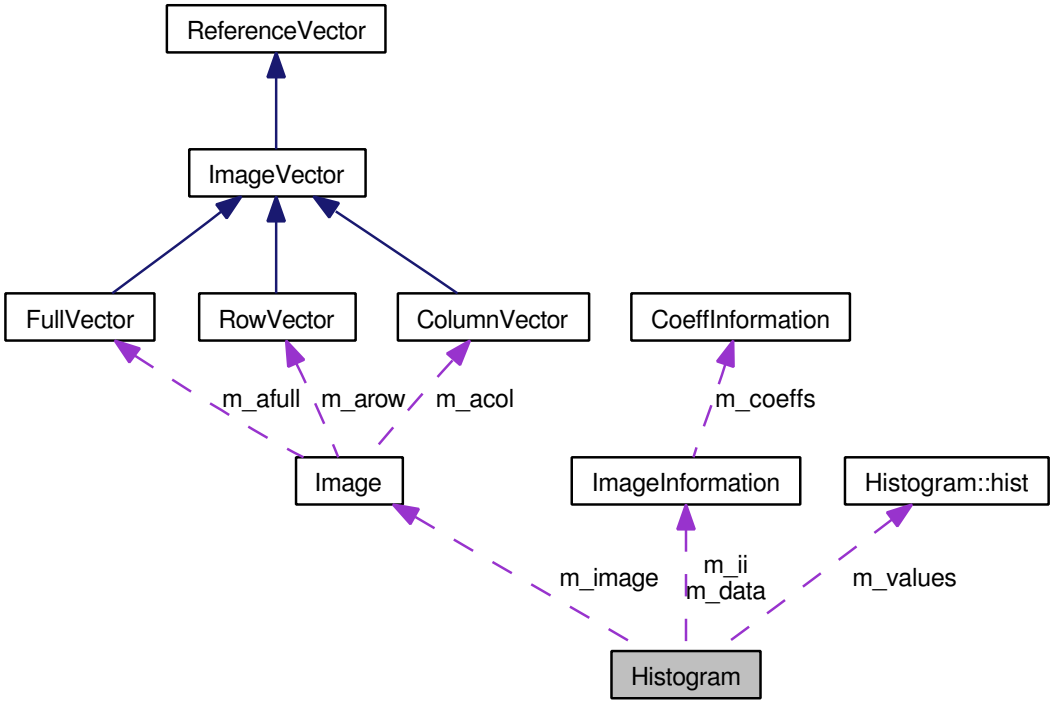
The documentation for this class was generated from the following file:

- GreymapWriter.hh

8.25 Histogram Class Reference

`#include <Histogram.hh>`

Collaboration diagram for Histogram:



Public Member Functions

- Histogram (Image &img, double dlt=1.0)
- Histogram (ImageInformation &ii, double dlt=1.0)
- ~Histogram (void)
- int at (int pos, coeff &lower)
- int at (coeff value)
- int size (void)
- double delta (void)
- void update (void)
- void print (void)
- void add (coeff value, int number)

Protected Attributes

- Image * m_image
- ImageInformation * m_data

- ImageInformation * m_ii
- double m_dlt
- struct hist * m_values
- int m_nvals

Classes

- struct hist

8.25.1 Detailed Description

A class for histograms.

Definition at line 21 of file Histogram.hh.

8.25.2 Constructor & Destructor Documentation

8.25.2.1 Histogram::Histogram (Image & *img*, double *dlt* = 1.0)

Constructor. Creates the histogram from an Image object.

Exceptions:

invalid_argument a zero value for {*dlt*} was specified

Parameters:

img the image to look at

dlt the quantization step (necessary for histograms in the DWT domain

8.25.2.2 Histogram::Histogram (ImageInformation & *ii*, double *dlt* = 1.0)

Constructor. Creates the histogram from an ImageInformation object.

Exceptions:

invalid_argument a zero value for {*dlt*} was specified

Parameters:

ii the image data to look at (will be duplicated)

dlt the quantization step (necessary for histograms in the DWT domain

8.25.2.3 Histogram::~Histogram (void)

Destructor. Releases allocated memory

8.25.3 Member Function Documentation

8.25.3.1 int Histogram::at (int *pos*, coeff & *lower*)

Get a value via an index.

Exceptions:

invalid_argument the index {*pos*} is out of bounds.

Parameters:

pos The index position.
lower The lower greyscale boundary (returned value).

Returns:

the value

8.25.3.2 int Histogram::at (coeff *value*)

Get a value via a greyscale value.

Parameters:

value the greyscale value.

Returns:

the value

8.25.3.3 int Histogram::size (void) [inline]

Return the number of numbers stored.

Returns:

the number of numbers.

Definition at line 59 of file Histogram.hh.

References m_nvals.

8.25.3.4 double Histogram::delta (void) [inline]

Get the histogram's delta.

Returns:

the delta

Definition at line 62 of file Histogram.hh.

References m_dlt.

8.25.3.5 void Histogram::update (void)

Update the Histogram. This may be necessary after changes to the image.

8.25.3.6 void Histogram::print (void)

Print the histogram to stdout.

8.25.3.7 void Histogram::add (coeff *value*, int *number*)

Add an occurrence number for a given value.

Exceptions:

invalid_argument adding number would make the result negative

Parameters:

value the value

number the number to be added

8.25.4 Member Data Documentation

8.25.4.1 Image* Histogram::m_image [protected]

A pointer as a reference to the image.

Definition at line 78 of file Histogram.hh.

8.25.4.2 ImageInformation* Histogram::m_data [protected]

A pointer as a reference to the ImageInformation object (needs to be stored in order to be able to update the histogram after changes to the original image).

Definition at line 82 of file Histogram.hh.

8.25.4.3 ImageInformation* Histogram::m_ii [protected]

A helper object for getting the image's coefficients sorted.

Definition at line 84 of file Histogram.hh.

8.25.4.4 double Histogram::m_dlt [protected]

The delta between two histogram slots.

Definition at line 86 of file Histogram.hh.

Referenced by delta().

8.25.4.5 struct hist* Histogram::m_values [read, protected]

The internal histogram (an array of hist structs).

Definition at line 97 of file Histogram.hh.

8.25.4.6 int Histogram::m_nvals [protected]

The number of histogram slots.

Definition at line 99 of file Histogram.hh.

Referenced by size().

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

- Histogram.hh

8.26 Histogram::hist Struct Reference

`#include <Histogram.hh>`

Public Attributes

- coeff lower
- coeff upper
- int number

8.26.1 Detailed Description

An internal structure for the histogram

Definition at line 88 of file Histogram.hh.

8.26.2 Member Data Documentation

8.26.2.1 coeff Histogram::hist::lower

The lower bound (value > lower)

Definition at line 90 of file Histogram.hh.

8.26.2.2 coeff Histogram::hist::upper

The upper bound (value >= lower)

Definition at line 92 of file Histogram.hh.

8.26.2.3 int Histogram::hist::number

The number of occurrences

Definition at line 94 of file Histogram.hh.

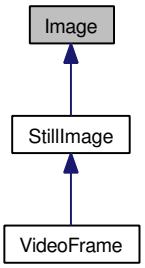
The documentation for this struct was generated from the following file:

- Histogram.hh

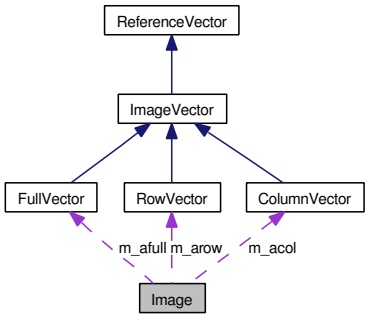
8.27 Image Class Reference

`#include <Image.hh>`

Inheritance diagram for Image:



Collaboration diagram for Image:



Public Member Functions

- Image (void)
- virtual ~Image (void)
- int rows (void) const
- int cols (void) const
- int size (void) const
- ColumnVector & col (int x=0)
- RowVector & row (int y=0)
- FullVector & full (int xy=0)
- virtual coeff at (int y, int x) const =0
- virtual coeff at (int abs) const =0
- virtual void to (int y, int x, coeff val)=0
- virtual void to (int abs, coeff val)=0
- virtual int abs (int y, int x) const =0
- virtual bool epsilons (Image &img, coeff epsilon) const =0
- bool epsilonsAt (int y, int x, coeff value, coeff epsilon) const

- bool epsilonsAt (int abs, coeff value, coeff epsilon) const
- virtual bool equals (Image &img) const =0
- virtual double iproduct (Image &img) const
- virtual void normalize (coeff factor)
- virtual void unnormalize (coeff factor)
- virtual void gammaCorrection (double factor, int yoffs=0, int xoffs=0, int ysize=-1, int xsize=-1, int norm=255)
- double gammaCorrectionAuto (int rows, int columns, int yoffs=0, int xoffs=0, int ysize=-1, int xsize=-1, int norm=255)
- virtual void histEqualization (int yoffs=0, int xoffs=0, int ysize=-1, int xsize=-1, int startH=0, int endH=255)
- virtual void beautify (int yoffs=0, int xoffs=0, int ysize=0, int xsize=0)
- virtual void truncate (coeff min=0, coeff max=255)
- virtual void pixelize (void)
- void valadjust (void)
- virtual void read (char const *fname, int rawy=0, int rawx=0)=0
- virtual void read (char const *fname, filetype ftype, int rawy=0, int rawx=0)=0
- virtual void write (char const *fname, bool beautify=false)=0
- virtual void write (char const *fname, filetype ftype, bool beautify=false)=0
- virtual void importCoeffs (int rows, int cols, coeff *buf)
- virtual coeff * exportCoeffs (void)
- virtual void importPixels (int rows, int cols, pixel *buf)
- virtual pixel * exportPixels (void) const
- virtual Image * clone (void) const =0
- virtual void copy (Image &img)
- Image * crop (int yoffs, int xoffs, int ysize, int xsize) const
- virtual void paste (int yoffs, int xoffs, Image &img)
- virtual void shift (int yoffs, int xoffs)
- int anasteps (void) const
- int synsteps (void) const
- void anasteps (int steps)
- void synsteps (int steps)
- void fill (coeff value, int fromY=0, int fromX=0, int toY=-1, int toX=-1)
- virtual coeff smax (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const =0
- virtual coeff smin (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const =0
- virtual coeff amax (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const =0
- virtual coeff amin (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const =0
- virtual coeff saverage (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const =0
- virtual coeff aaverage (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const =0
- virtual coeff sqvariance (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const
- virtual coeff variance (int fromY, int fromX, int toY, int toX, coeff avg, bool abs) const
- virtual coeff variance (int fromY=0, int fromX=0, int toY=-1, int toX=-1, bool abs=false) const

- virtual coeff sdeviation (int fromY=0, int fromX=0, int toY=-1, int toX=-1, bool abs=false) const
- virtual void resize (int rows, int cols)=0
- virtual Image * fitInto (int rows, int cols, int fill=-1, int function=0) const
- virtual Image * scale (double factor, int function=0) const
- virtual Image * scale (double yFactor, double xFactor, int function=0) const

Protected Member Functions

- void dimcheck (const Image &img) const
- virtual Image * mkImage (int rows=0, int cols=0) const =0

Protected Attributes

- ColumnVector * m_acol
- RowVector * m_arow
- FullVector * m_afull
- int m_ysize
- int m_xsize
- int m_xysize
- int m_asteps
- int m_ssteps

8.27.1 Detailed Description

An image (abstract). A (grey-scale) more-than-one-dimensional image.

Definition at line 29 of file Image.hh.

8.27.2 Constructor & Destructor Documentation

8.27.2.1 Image::Image (void)

Constructor. Only sets standard values.

8.27.2.2 virtual Image::~Image (void) [inline, virtual]

Destructor. Does nothing.

Definition at line 37 of file Image.hh.

8.27.3 Member Function Documentation

8.27.3.1 int Image::rows (void) const [inline]

Return the number of rows.

Returns:

the number of rows

Definition at line 42 of file Image.hh.

Referenced by StillImage::makeEmpty().

8.27.3.2 int Image::cols (void) const [inline]

Return the number of columns.

Returns:

the number of columns

Definition at line 46 of file Image.hh.

Referenced by StillImage::makeEmpty().

8.27.3.3 int Image::size (void) const [inline]

Return the overall size.

Returns:

the overall size

Definition at line 50 of file Image.hh.

8.27.3.4 ColumnVector& Image::col (int *x* = 0) [inline]

Return a column reference. Returns a pseudo-vector that provides a reference to one column of the image.

Parameters:

x The number of the column to be returned

Returns:

the column reference

Definition at line 58 of file Image.hh.

8.27.3.5 RowVector& Image::row (int *y* = 0) [inline]

Return a row reference. Returns a pseudo-vector that provides a reference to one row of the image.

Parameters:

y The number of the row to be returned

Returns:

the row reference

Definition at line 65 of file Image.hh.

8.27.3.6 FullVector& Image::full (int *xy* = 0) [inline]

Return a full reference. Returns a pseudo-vector that provides a reference to the image as a big vector.

Parameters:

xy The position where to start

Returns:

the full reference

Definition at line 73 of file Image.hh.

8.27.3.7 virtual coeff Image::at (int *y*, int *x*) const [pure virtual]

Get a value. Returns the value at row {*y*} and col {*x*}.

Parameters:

y the row

x the col

Returns:

the value

Implemented in StillImage.

8.27.3.8 virtual coeff Image::at (int *abs*) const [pure virtual]

Get a value. Returns the value at absolute position {*abs*}.

Parameters:

abs the position

Returns:

the value

Implemented in StillImage.

8.27.3.9 virtual void Image::to (int *y*, int *x*, coeff *val*) [pure virtual]

Set a value. Sets the value at row {*y*} and col {*x*}.

Parameters:

y the row
x the col
val the new value

Returns:

the value

Implemented in StillImage.

8.27.3.10 virtual void Image::to (int *abs*, coeff *val*) [pure virtual]

Set a value. Sets the value at absolute position {*abs*}.

Parameters:

abs the position
val the new value

Implemented in StillImage.

8.27.3.11 virtual int Image::abs (int *y*, int *x*) const [pure virtual]

Return the absolute offset of a position.

Parameters:

y the position's row
x the position's col

Returns:

the absolute offset

Implemented in StillImage.

8.27.3.12 virtual bool Image::epsilons (Image & *img*, coeff *epsilon*) const [pure virtual]

Rough comparison. See if two images are similar according to a given {*epsilon*} (important for floating-point comparisons).

Parameters:

img the other {*Image*} object

epsilon the epsilon

Returns:

if both are identical: {*true*}, else {*false*}

Implemented in StillImage.

8.27.3.13 bool Image::epsilonsAt (int *y*, int *x*, coeff *value*, coeff *epsilon*) const

Rough comparison. See if an image value is equal to some value according to a given {*epsilon*} (important for floating-point comparisons).

Parameters:

y the row

x the col

value the value to compare to

epsilon the epsilon

Returns:

if both are identical: {*true*}, else {*false*}

8.27.3.14 bool Image::epsilonsAt (int *abs*, coeff *value*, coeff *epsilon*) const

Rough comparison. See if an image value is equal to some value according to a given {*epsilon*} (important for floating-point comparisons).

Parameters:

abs the absolute position

value the value to compare to

epsilon the epsilon

Returns:

if both are identical: {*true*}, else {*false*}

8.27.3.15 virtual bool Image::equals (Image & *img*) const [pure virtual]

Compares two images. Return {*true*} if both are equal.

Parameters:

img The other {*Image*} object

Returns:

if equals: {*true*}, else {*false*}

Implemented in StillImage.

8.27.3.16 virtual double Image::iproduct (Image & *img*) const [virtual]

Computes the inner product between this and another image. Both images have to have identical dimensions.

Exceptions:

invalid_argument The images don't have identical dimensions

Parameters:

img The other {*Image*} object

Returns:

the inner product

8.27.3.17 virtual void Image::normalize (coeff *factor*) [virtual]

Normalize coefficients.

Parameters:

factor the normalization factor

8.27.3.18 virtual void Image::unnormalize (coeff *factor*) [virtual]

Unnormalize coefficients.

Parameters:

factor the unnormalization factor

8.27.3.19 virtual void Image::gammaCorrection (double *factor*, int *yoffs* = 0, int *xoffs* = 0, int *ysize* = -1, int *xsize* = -1, int *norm* = 255) [virtual]

Perform a gamma correction. Since this is normally only defined for the spatial domain (the pixels need to be normalized) the normalization factor is set to 255 by default, if values are found in the image outside the specified [0..norm] range, the range will be increased. If the range is increased a new normalization factor is automatically calculated as the difference between the image's maximum (if more than norm) and minimum (if less than 0) coefficient value.

Parameters:

factor the gamma factor (0..1.0)

ysize the horizontal size of the region to process

xsize the vertical size of the region to process

yoffs the horizontal offset of the region to process

xoffs the vertical offset of the region to process

norm the normalization factor (default = 255)

8.27.3.20 double Image::gammaCorrectionAuto (int *rows*, int *columns*, int *yoffs* = 0, int *xoffs* = 0, int *ysize* = -1, int *xsize* = -1, int *norm* = 255)

Does a automatic gamma correction. This function breaks the image up into smaller matrices and then calculates their averages. The highest and lowest average is then used to calculate a gamma value which is used to a parameter in a call to gammaCorrection().

Note:

If the image can not be split into the required sub-areas without a remainder the nearest solution (down) is used.

Parameters:

rows Split image horizontly into subsections

columns Split image vertically into subsections

yoffs Vertical offset

xoffs Horizontal offset

ysize Vertical size

xsize Horizontal size

norm the normalization factor (default = 255)

Returns:

the gamma value - (use it to undo operation if needed)

8.27.3.21 virtual void Image::histEqualization (int *yoffs* = 0, int *xoffs* = 0, int *ysize* = -1, int *xsize* = -1, int *startH* = 0, int *endH* = 255) [virtual]

Perform a histogram equalisation. The operation performs a floating-point to int quantisation and uses a given color range if 0..255 is not desired. Explicit values for the color range can be necessary e.g. for Wavelet-decomposed images; for example for Haar LL Level 1 a good choice would be 0..1023 and -512..511 for Haar non LL Level 1. Code provided by Johan Ehlers.

Warning:

Use unnormalised coefficients else the operation has un-determinable results.

Parameters:

ysize the horizontal size of the region to process

xsize the vertical size of the region to process

yoffs the horizontal offset of the region to process

xoffs the vertical offset of the region to process

startH the start value for range of histogram default = 0

endH the end value for the range of the histogram default = 255

8.27.3.22 virtual void Image::beautify (int *yoffs* = 0, int *xoffs* = 0, int *ysize* = 0, int *xsize* = 0) [virtual]

Do an intelligent kind of beautification considering previous transform steps. The original data will be overwritten, so this should only be used on clones or if the image is not going to be transformed afterwise. One region inside the image could be the low-pass subband and should thus be handled separately. If that is desired, the {*ysize}* and {*xsize}* parameters should be set to nonzero values.

Parameters:

yoffs the horizontal offset of the region that should be handled separately

xoffs the vertical offset of the region that should be handled separately

ysize the horizontal size of the region that should be handled separately

xsize the vertical size of the region that should be handled separately

8.27.3.23 virtual void Image::truncate (coeff *min* = 0, coeff *max* = 255) [virtual]

Do a dumb kind of beautification by just truncating values to their possible maxima depending on the image format or whatever. The original data will be overwritten, so this should only be used on clones or if the original data is not needed anymore.

Parameters:

min the minimum allowed value

max the maximum allowed value

8.27.3.24 virtual void Image::pixelize (void) [virtual]

First truncate the values to the [0..255] range, and then round them to integer numbers.

8.27.3.25 void Image::valadjust (void)

Adjust the coefficients to make them pixelizable.

8.27.3.26 virtual void Image::read (char const * *fname*, int *rawy* = 0, int *rawx* = 0) [pure virtual]

Read an image (abstract). Reads an image from a file guessing the file type from the file name's extension.

Parameters:

fname the file name, if {*NULL*}, then {*stdin*}

rawy the number of rows (only needed for RAW format)

rawx the number of cols (only needed for RAW format)

Implemented in StillImage.

8.27.3.27 virtual void Image::read (char const * *fname*, filetype *ftype*, int *rawy* = 0, int *rawx* = 0) [pure virtual]

Read an image (abstract). Reads an image from a file using the specified file type.

Parameters:

fname the file name, if {*NULL*}, then {*stdin*}

ftype the file type (image format)

rawy the number of rows (only needed for RAW format)

rawx the number of cols (only needed for RAW format)

Implemented in StillImage.

8.27.3.28 virtual void Image::write (char const * *fname*, bool *beautify* = false) [pure virtual]

Write an image (abstract). Writes an image to a file guessing the file type from the file name's extension.

Parameters:

fname the file name, if {*NULL*}, then {*stdout*}
beautify beautify images that have not had more analysis than synthesis steps?

Implemented in StillImage.

8.27.3.29 virtual void Image::write (char const * *fname*, filetype *ftype*, bool *beautify* = false) [pure virtual]

Write an image (abstract). Writes an image to a file using the specified file type.

Parameters:

ftype the file type
fname the file name, if {*NULL*}, then {*stdout*}
beautify beautify images that have not had more analysis than synthesis steps?

Implemented in StillImage.

8.27.3.30 virtual void Image::importCoeffs (int *rows*, int *cols*, coeff * *buf*) [virtual]

Import an image. The values will be copied.

Parameters:

rows the number of image rows (logical)
cols the number of image cols (logical)
buf the buffer holding the image coefficients (consists of rows * cols elements)

8.27.3.31 virtual coeff* Image::exportCoeffs (void) [virtual]

Exports an image.

Returns:

a new buffer holding the image coefficients (consists of rows * cols elements)

8.27.3.32 virtual void Image::importPixels (int *rows*, int *cols*, pixel * *buf*) [virtual]

Import a raw image. The values will be copied.

Parameters:

rows the number of image rows (logical)
cols the number of image cols (logical)
buf the buffer holding the image pixels (consists of rows * cols elements)

8.27.3.33 virtual pixel* Image::exportPixels (void) const [virtual]

Exports a raw image. Values exceeding the 0..255 range will be truncated.

Returns:

a new buffer holding the image pixels (consists of rows * cols elements)

8.27.3.34 virtual Image* Image::clone (void) const [pure virtual]

Produce a copy (abstract). Every dynamically object will be cloned rather than passing on the reference.

Returns:

the new, copied object.

Implemented in StillImage, and VideoFrame.

8.27.3.35 virtual void Image::copy (Image & *img*) [virtual]

Copy all data from a source image.

Parameters:

img the source image

8.27.3.36 Image* Image::crop (int *yoffs*, int *xoffs*, int *ysize*, int *xsize*) const

Returns a subimage.

Parameters:

yoffs the row where to start

xoffs the col where to start

ysize the vertical size

xsize the horizontal size

Returns:

the new image

8.27.3.37 virtual void Image::paste (int *yoffs*, int *xoffs*, Image & *img*) [virtual]

Inserts a subimage.

Parameters:

yoffs the row where to start

xoffs the col where to start

img the image to insert

8.27.3.38 virtual void Image::shift (int *yoffs*, int *xoffs*) [virtual]

Shift an Image.

Parameters:

yoffs the rows to shift

xoffs the cols to shift

8.27.3.39 int Image::anasteps (void) const [inline]

Returns the number of analysis steps performed so far.

Returns:

the number of steps

Definition at line 393 of file Image.hh.

8.27.3.40 int Image::synsteps (void) const [inline]

Returns the number of synthesis steps performed so far.

Returns:

the number of steps

Definition at line 397 of file Image.hh.

8.27.3.41 void Image::anasteps (int *steps*) [inline]

Sets the number of analysis steps performed so far.

Parameters:

steps the number of steps

Definition at line 401 of file Image.hh.

8.27.3.42 void Image::synsteps (int *steps*) [inline]

Sets the number of synthesis steps performed so far.

Parameters:

steps the number of steps

Definition at line 405 of file Image.hh.

8.27.3.43 void Image::fill (coeff *value*, int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1)

Fills an image (or regions of it) with a given value

Parameters:

val the new value for the selected positions

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

8.27.3.44 virtual coeff Image::smax (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const [pure virtual]

Returns the maximum value in a region. Signs will be considered.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the maximum

Implemented in StillImage.

8.27.3.45 virtual coeff Image::smin (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const [pure virtual]

Returns the maximum value in a region. Signs will be considered.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the maximum

Implemented in StillImage.

8.27.3.46 virtual coeff Image::amax (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const [pure virtual]

Returns the minimum absolute value in a region. Signs will be discarded.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the maximum

Implemented in StillImage.

8.27.3.47 virtual coeff Image::amin (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const [pure virtual]

Returns the minimum absolute value in a region. Signs will be discarded.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the minimum

Implemented in StillImage.

8.27.3.48 virtual coeff Image::saverage (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const [pure virtual]

Return the average color for a rectangular region inside the image drawn from one point within and the second point just outside the region. The coefficients' signedness will be considered.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the average color

Implemented in StillImage.

8.27.3.49 virtual coeff Image::aaverage (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const [pure virtual]

Return the average color for a rectangular region inside the image drawn from one point within and the second point just outside the region. The coefficients' signedness will be discarded.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the average color

Implemented in StillImage.

8.27.3.50 virtual coeff Image::sqvariance (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const [virtual]

Return the square variance for a rectangular region inside the image drawn from one point within and the second point just outside the region. Signs will be considered. Use this method if you want statistics on a region in the image. For other cases you may want to use ImageInformation::sqvariance().

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the square variance

8.27.3.51 virtual coeff Image::variance (int *fromY*, int *fromX*, int *toY*, int *toX*, coeff *avg*, bool *abs*) const [virtual]

Return the variance for a rectangular region inside the image drawn from one point within and the second point just outside the region. Use this method if you want statistics on a region in the image. For other cases you may want to use ImageInformation::variance(). This version of the method takes the average (either absolute or signed) as argument so that it does not need to be calculated again.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

avg the average for the region

abs true if signs are discarded

Returns:

the square variance

8.27.3.52 virtual coeff Image::variance (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1, bool *abs* = false) const [virtual]

Return the variance for a rectangular region inside the image drawn from one point within and the second point just outside the region. Use this method if you want statistics on a region in the image. For other cases you may want to use ImageInformation::variance().

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

abs true if signs are discarded

Returns:

the square variance

8.27.3.53 virtual coeff Image::sdeviation (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1, bool *abs* = false) const [virtual]

Returns the standard deviation for a rectangular region inside the image drawn from one point within and the second point just outside the region. Signs will be considered. Use this method if you want statistics on a region in the image. For other cases you may want to use ImageInformation::sdeviation().

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

abs true if signs are discarded

Returns:

the standard deviation.

8.27.3.54 virtual void Image::resize (int *rows*, int *cols*) [pure virtual]

Resize the image's x/y dimensions. The old values will remain as far as they fit in the new dimensions. The new dimensions must all be greater than zero.

Exceptions:

invalid_argument one or both dimensions are either negative or zero

Parameters:

rows the new number of rows

cols the new number of cols

Implemented in StillImage.

8.27.3.55 virtual Image* Image::fitInto (int *rows*, int *cols*, int *fill* = -1, int *function* = 0) const [virtual]

Produce a scaled version of the image which just fits into the given dimensions. If the aspect ratio does not fit into the new dimensions the remaining areas can be filled with a given color or alternatively the resulting image will be smaller than the target dimensions keeping its aspect ratio.

Exceptions:

invalid_argument the factor is negative or the function argument is invalid

Parameters:

rows the target number of rows

cols the target number of columns

fill if not less than zero use this to fill remaining space (else the a smaller image size will be chosen if the aspect ratio does not match)

function interpolation function (0: bilinear interpolation, 1: average, 2: nearest neighbour).

Returns:

a new rescaled image

8.27.3.56 virtual Image* Image::scale (double *factor*, int *function* = 0) const [virtual]

Produce a scaled version of the image. The aspect ratio will remain the same.

Exceptions:

invalid_argument the factor is negative or the function argument is invalid

Parameters:

factor the scale factor (>= 0, 1 for no change)

function interpolation function (0: bilinear interpolation, 1: average, 2: nearest neighbour).

Returns:

a new rescaled image

8.27.3.57 virtual Image* Image::scale (double *yFactor*, double *xFactor*, int *function* = 0) const [virtual]

Produce a scaled version of the image. The aspect ratio depends on the two scale factors

Exceptions:

invalid_argument the factor is negative or the function argument is invalid

Parameters:

yFactor the vertical scale factor (>= 0, 1 for no change)

xFactor the horizontal scale factor (>= 0, 1 for no change)

function interpolation function (0: bilinear interpolation, 1: average, 2: nearest neighbour).

Returns:

a new rescaled image

8.27.3.58 void Image::dimcheck (const Image & *img*) const [protected]

Make sure the other image's dimensions match.

Exceptions:

invalid_argument The other image's dimensions differ

Parameters:

img the other image

8.27.3.59 virtual Image* Image::mkImage (int *rows* = 0, int *cols* = 0) const [protected, pure virtual]

Abstract factory method to get a new instance of a given size filled with zeroes.

Parameters:

rows the number of rows

cols the number of cols

Returns:

the new image

Implemented in StillImage.

8.27.4 Member Data Documentation

8.27.4.1 ColumnVector* Image::m_acol [protected]

Column reference. A vector that can be set to any column to allow iterating over its values.

Definition at line 605 of file Image.hh.

8.27.4.2 RowVector* Image::m_arow [protected]

Row reference. A vector that can be set to any row to allow iterating over its values.

Definition at line 608 of file Image.hh.

8.27.4.3 FullVector* Image::m_afull [protected]

Full reference. A vector that can be set anywhere within the pixels to iterating over its values.

Definition at line 611 of file Image.hh.

8.27.4.4 int Image::m_ysize [protected]

Number of rows.

Definition at line 613 of file Image.hh.

8.27.4.5 int Image::m_xsize [protected]

Number of cols.

Definition at line 615 of file Image.hh.

8.27.4.6 int Image::m_xysize [protected]

Overall size.

Definition at line 617 of file Image.hh.

8.27.4.7 int Image::m_asteps [protected]

The number of steps performed in the last analysis.

Definition at line 619 of file Image.hh.

8.27.4.8 int Image::m_ssteps [protected]

The number of steps performed in the last synthesis.

Definition at line 621 of file Image.hh.

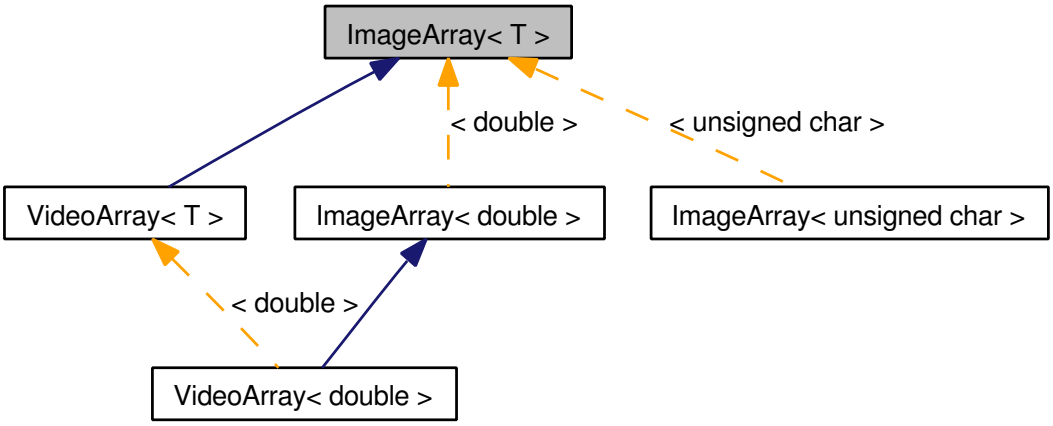
The documentation for this class was generated from the following file:

- Image.hh

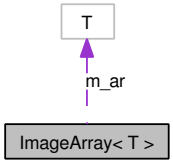
8.28 ImageArray< T > Class Template Reference

`#include <ImageArray.hh>`

Inheritance diagram for ImageArray< T >:



Collaboration diagram for ImageArray< T >:



Public Member Functions

- ImageArray (void)
- ImageArray (int rows, int cols)
- virtual ~ImageArray (void)
- int rows (void) const
- int cols (void) const
- int size (void) const
- T at (int y, int x) const
- virtual T at (int abs) const
- void to (int y, int x, T val)
- virtual void to (int abs, T val)
- virtual int abs (int y, int x) const
- T smax (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const
- T smin (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const
- T amax (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const

- T amin (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const
- T saverage (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const
- T aaverage (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const
- virtual void resize (int rows, int cols)
- virtual void import (int rows, int cols, T *array)
- virtual ImageArray< T > * clone (void) const
- virtual void copy (ImageArray< T > &ia)
- void copy (ImageArray< T > *ia)
- virtual bool epsilons (ImageArray< T > &ia, T epsilon) const
- bool epsilons (ImageArray< T > *ia, T epsilon) const
- bool equals (ImageArray< T > &ia) const
- bool equals (ImageArray< T > *ia) const
- T *& array (void)

Protected Member Functions

- virtual void updateRowsArray (void)

Protected Attributes

- int m_xsize
- int m_ysize
- int m_xysize
- T * m_ar
- int * m_rows

8.28.1 Detailed Description

template<class T> class ImageArray< T >

A two-dimensional array wrapper. This allows us to access the image's pixels or coefficients and provides some utility methods. It is the main working horse for the Image framework.

Definition at line 23 of file ImageArray.hh.

8.28.2 Constructor & Destructor Documentation

8.28.2.1 template<class T> ImageArray< T >::ImageArray (void)

Constructor. Sets the dimension to zero.

8.28.2.2 template<class T> ImageArray< T >::ImageArray (int *rows*, int *cols*)

Constructor. Sets the dimension and allocates space. A size of zero is allowed, negative values not.

Parameters:

rows the vertical size

cols the horizontal size

8.28.2.3 template<class T> virtual ImageArray< T >::~ImageArray (void) [virtual]

Destructor. Releases allocated memory.

8.28.3 Member Function Documentation

8.28.3.1 template<class T> int ImageArray< T >::rows (void) const [inline]

Returns the vertical size.

Returns:

the vertical size

Definition at line 43 of file ImageArray.hh.

8.28.3.2 template<class T> int ImageArray< T >::cols (void) const [inline]

Returns the horizontal size.

Returns:

the horizontal size

Definition at line 47 of file ImageArray.hh.

8.28.3.3 template<class T> int ImageArray< T >::size (void) const [inline]

Returns the overall size (x∗y).

Returns:

the overall size

Definition at line 51 of file ImageArray.hh.

8.28.3.4 template<class T> T ImageArray< T >::at (int *y*, int *x*) const [inline]

Returns the value at (x,y).

Parameters:

y the row

x the col

Returns:

the value

Reimplemented in VideoArray< T >, and VideoArray< double >.

Definition at line 59 of file ImageArray.hh.

8.28.3.5 template<class T> virtual T ImageArray< T >::at (int *abs*) const [inline, virtual]

Returns the array's {*n*}*th* value.

Parameters:

abs the offset from the array start

Returns:

the value

Reimplemented in VideoArray< T >, and VideoArray< double >.

Definition at line 66 of file ImageArray.hh.

8.28.3.6 template<class T> void ImageArray< T >::to (int *y*, int *x*, T *val*) [inline]

Sets the value at (x,y).

Parameters:

y the row

x the col

val the value

Reimplemented in VideoArray< T >, and VideoArray< double >.

Definition at line 74 of file ImageArray.hh.

8.28.3.7 template<class T> virtual void ImageArray< T >::to (int *abs*, T *val*) [inline, virtual]

Sets the array's {*n*}*th* value.

Parameters:

abs the offset from the array start

val the value

Reimplemented in VideoArray< T >, and VideoArray< double >.

Definition at line 80 of file ImageArray.hh.

8.28.3.8 template<class T> virtual int ImageArray< T >::abs (int *y*, int *x*) const [inline, virtual]

Return the absolute offset of a position.

Parameters:

y the position's row

x the position's col

Returns:

the absolute offset

Reimplemented in VideoArray< T >, and VideoArray< double >.

Definition at line 89 of file ImageArray.hh.

Referenced by ImageArray< unsigned char >::at(), and ImageArray< unsigned char >::to().

8.28.3.9 template<class T> T ImageArray< T >::smax (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const

Returns the maximum value in a region. Signs will be considered.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the maximum

8.28.3.10 template<class T> T ImageArray< T >::smin (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const

Returns the minimum value in a region. Signs will be considered.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the minimum

8.28.3.11 template<class T> T ImageArray< T >::amax (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const

Returns the maximum absolute value in a region. Signs will be discarded.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the maximum

8.28.3.12 template<class T> T ImageArray< T >::amin (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const

Returns the minimum absolute value in a region. Signs will be discarded.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the minimum

8.28.3.13 template<class T> T ImageArray< T >::saverage (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const

Return the average color for a rectangular region inside the image drawn from one point within and the second point just outside the region. The coefficients' signedness will be considered.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the average color

8.28.3.14 template<class T> T ImageArray< T >::aaverage (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const

Return the average color for a rectangular region inside the image drawn from one point within and the second point just outside the region. The coefficients' signedness will be discarded.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the average color

8.28.3.15 template<class T> virtual void ImageArray< T >::resize (int *rows*, int *cols*) [virtual]

Resize the array. The old values will be copied to the new dimension as far as they fit in. The new dimensions must all be greater than zero.

Exceptions:

invalid_argument one or both dimensions are either negative or zero

Parameters:

rows the new number of rows

cols the new number of cols

Reimplemented in VideoArray< T >, and VideoArray< double >.

8.28.3.16 template<class T> virtual void ImageArray< T >::import (int *rows*, int *cols*, T * *array*) [virtual]

Import a raw array. The new dimensions and the new array will be stored discarding the old ones.

Parameters:

rows the new number of rows

cols the new number of cols

array the new array

Reimplemented in VideoArray< T >, and VideoArray< double >.

8.28.3.17 template<class T> virtual ImageArray<T>* ImageArray< T >::clone (void) const [virtual]

Create a copy. All values will be duplicated rather than the references.

Returns:

the new object

Reimplemented in VideoArray< T >, and VideoArray< double >.

8.28.3.18 template<class T> virtual void ImageArray< T >::copy (ImageArray< T > & *ia*) [virtual]

Copy from a different array.

Parameters:

ia the other array

Reimplemented in VideoArray< T >, and VideoArray< double >.

Referenced by ImageArray< unsigned char >::copy().

8.28.3.19 template<class T> void ImageArray< T >::copy (ImageArray< T > * *ia*) [inline]

Copy from a different array.

Parameters:

ia the other array

Definition at line 173 of file ImageArray.hh.

8.28.3.20 template<class T> virtual bool ImageArray< T >::epsilons (ImageArray< T > & *ia*, T *epsilon*) const [virtual]

Rough comparison. See if two arrays are similar according to a given {*epsilon*} (important for floating-point comparisons).

Parameters:

ia the other {*ImageArray*} object

epsilon the epsilon

Returns:

if both are identical: {*true*}, else {*false*}

Reimplemented in VideoArray< T >, and VideoArray< double >.

Referenced by ImageArray< unsigned char >::epsilons(), and ImageArray< unsigned char >::equals().

8.28.3.21 template<class T> bool ImageArray< T >::epsilons (ImageArray< T > * *ia*, T *epsilon*) const [inline]

Rough comparison. See if two arrays are similar according to a given {*epsilon*} (important for floating-point comparisons).

Parameters:

ia the other {*ImageArray*} object

epsilon the epsilon

Returns:

if both are identical: {*true*}, else {*false*}

Definition at line 191 of file ImageArray.hh.

8.28.3.22 template<class T> bool ImageArray< T >::equals (ImageArray< T > & *ia*) const [inline]

Exact comparison. See if two arrays are similar.

Parameters:

ia the other {*ImageArray*} object

Returns:

if both are identical: {*true*}, else {*false*}

Definition at line 198 of file ImageArray.hh.

Referenced by ImageArray< unsigned char >::equals().

8.28.3.23 template<class T> bool ImageArray< T >::equals (ImageArray< T > * *ia*) const [inline]

Exact comparison. See if two arrays are similar.

Parameters:

ia the other {*ImageArray*} object

Returns:

if both are identical: {*true*}, else {*false*}

Definition at line 205 of file ImageArray.hh.

8.28.3.24 template<class T> T*& ImageArray< T >::array (void) [inline]

Physical access to the array. This is dangerous but necessary for performance reasons because we sometimes need to fill the array with more than one value at a time.

Returns:

a reference to the array

Definition at line 212 of file ImageArray.hh.

8.28.3.25 template<class T> virtual void ImageArray< T >::updateRowsArray (void) [protected, virtual]

Allocates and sets the 'm_rows' member (e.g. after a resize or in the constructor). If 'm_rows' is not NULL it will be deleted first

8.28.4 Member Data Documentation

8.28.4.1 template<class T> int ImageArray< T >::m_xsize [protected]

The array's horizontal size.

Definition at line 216 of file ImageArray.hh.

Referenced by ImageArray< unsigned char >::cols().

8.28.4.2 template<class T> int ImageArray< T >::m_ysize [protected]

The array's vertical size.

Definition at line 218 of file ImageArray.hh.

Referenced by ImageArray< unsigned char >::rows().

8.28.4.3 template<class T> int ImageArray< T >::m_xysize [protected]

The array's overall size (x∗y).

Definition at line 220 of file ImageArray.hh.

Referenced by ImageArray< unsigned char >::size().

8.28.4.4 template<class T> T∗ ImageArray< T >::m_ar [protected]

The actual array.

Definition at line 222 of file ImageArray.hh.

Referenced by ImageArray< unsigned char >::array(), VideoArray< double >::at(), ImageArray< unsigned char >::at(), VideoArray< double >::to(), and ImageArray< unsigned char >::to().

8.28.4.5 template<class T> int∗ ImageArray< T >::m_rows [protected]

An array holding all rows' offsets (optimizes addressing)

Definition at line 224 of file ImageArray.hh.

Referenced by VideoArray< double >::abs(), and ImageArray< unsigned char >::abs().

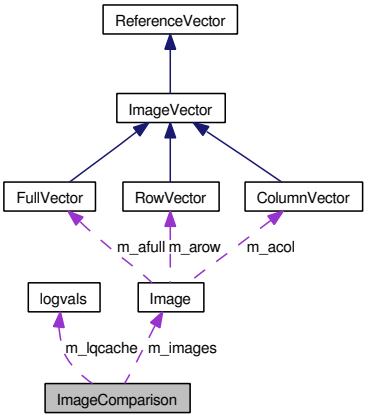
The documentation for this class was generated from the following file:

- ImageArray.hh

8.29 ImageComparison Class Reference

`#include <ImageComparison.hh>`

Collaboration diagram for ImageComparison:



Public Member Functions

- ImageComparison (const Image &img1, const Image &img2)
- ~ImageComparison (void)
- int rows (void) const
- int cols (void) const
- int size (void) const
- Image * difference (coeff magnify=1, bool reverse=false)
- double cmpsnr (void)
- double cmpmse (void)
- lq distlq (double percent, FilterSet &flt, imgtype type)
- double distlqd (double percent, FilterSet &flt, imgtype type)

Protected Member Functions

- int level (int pos)
- double weight (int y, int x, imgtype type)
- double lq_sum (imgtype type, bool reverse, bool oldLqd=false)

Protected Attributes

- int m_ysize
- int m_xsize
- int m_xysize
- Image * m_images [2]

- logvals * m_lqcache
- double m_weights [2][6]
- int m_lqcachesize

8.29.1 Detailed Description

A class for differences between two images

Definition at line 40 of file ImageComparison.hh.

8.29.2 Constructor & Destructor Documentation

8.29.2.1 ImageComparison::ImageComparison (const Image & *img1*, const Image & *img2*)

Constructor. Creates copies of the two images. If the two images have different dimensions they will be resized to the largest common dimensions.

Parameters:

img1 the first image

img2 the second image

8.29.2.2 ImageComparison::~ImageComparison (void)

Destructor. Releases allocated memory.

8.29.3 Member Function Documentation

8.29.3.1 int ImageComparison::rows (void) const [inline]

Return the number of rows.

Returns:

the number of rows

Definition at line 57 of file ImageComparison.hh.

References m_ysize.

8.29.3.2 int ImageComparison::cols (void) const [inline]

Return the number of columns.

Returns:

the number of columns

Definition at line 61 of file ImageComparison.hh.

References m_xsize.

8.29.3.3 int ImageComparison::size (void) const [inline]

Return the overall size.

Returns:

the overall size

Definition at line 65 of file ImageComparison.hh.

References m_xysize.

8.29.3.4 Image* ImageComparison::difference (coeff *magnify* = 1, bool *reverse* = false)

Produce an image holding the difference between the first and the second image

Parameters:

magnify multiply the result by a factor to increase / decrease visibility

reverse reverse the subtraction (take img1 - img2)

Returns:

a new image containing the difference

8.29.3.5 double ImageComparison::cmpsnr (void)

Compare with another image producing the PSNR. Stolen from Geoff Davis' Wavelet Coder kit.

Returns:

The PSNR

8.29.3.6 double ImageComparison::cmpmse (void)

Compare with another image producing the mean square error. Stolen from Geoff Davis' Wavelet Coder kit.

Returns:

The MSE

8.29.3.7 lq ImageComparison::distlq (double *percent*, FilterSet & *flt*, imgtype *type*)

Calculate the distance to another image according to the L^q norm. Idea and basic algorithm from Jacobs, Finkelstein, Salesin: "Fast multiresolution image querying", Proc. of SIGGRAPH, 95.

Exceptions:

invalid_argument image is not square

Parameters:

percent the relative number of Wavelet coefficients to compare

flt the filter to use

type the type of the images to be assumed (drawn or scanned)

Returns:

The L^q norm

8.29.3.8 double ImageComparison::distlqd (double *percent*, FilterSet & *flt*, imgtype *type*)

Calculate the distance to another image according to the old L^qd norm. Idea and basic algorithm from Jacobs, Finkelstein, Salesin: "Fast multiresolution image querying", Proc. of SIGGRAPH, 95.

Exceptions:

invalid_argument image is not square

Parameters:

percent the relative number of Wavelet coefficients to compare

flt the filter to use

type the type of the images to be assumed (drawn or scanned)

Returns:

The L^qd norm

8.29.3.9 int ImageComparison::level (int *pos*) [protected]

Helper method for distlq(). Calculates the weighing level for a position

Parameters:

pos the position (row or col)

Returns:

the level

8.29.3.10 double ImageComparison::weight (int *y*, int *x*, imgtype *type*) [protected]

Helper method for distlq(). Calculate the weight for a position (y, x).

Parameters:

y the row

x the col

type the images' type to be assumed (drawn or scanned)

Returns:

the position's weight

8.29.3.11 double ImageComparison::lq_sum (imgtype *type*, bool *reverse*, bool *oldLqd* = false) [protected]

Helper method for distlq(). Calculate the sum of differences of the two decomposed images.

Parameters:

type the type (drawn or scanned) to be assumed

reverse if true, subract the first from the second image

oldLqd if true, calculate the old Lqd way: add up the score for differing locations instead of subtracting the scores for identical values

8.29.4 Member Data Documentation

8.29.4.1 int ImageComparison::m_ysize [protected]

Number of rows.

Definition at line 123 of file ImageComparison.hh.

Referenced by rows().

8.29.4.2 int ImageComparison::m_xsize [protected]

Number of cols.

Definition at line 125 of file ImageComparison.hh.

Referenced by cols().

8.29.4.3 int ImageComparison::m_xysize [protected]

Overall size.

Definition at line 127 of file ImageComparison.hh.

Referenced by size().

8.29.4.4 Image* ImageComparison::m_images[2] [protected]

An array holding the two images to be compared.

Definition at line 130 of file ImageComparison.hh.

8.29.4.5 logvals* ImageComparison::m_lqcache [protected]

A cache needed by distlq() to avoid repeated calculation of floor (log (i) / log (2)).

Definition at line 134 of file ImageComparison.hh.

8.29.4.6 double ImageComparison::m_weights[2][6] [protected]

The six weights for scanned images, needed by distlq(), Y channel, according to Jacobs et.al.

Definition at line 138 of file ImageComparison.hh.

8.29.4.7 int ImageComparison::m_lqcachesize [protected]

The size of the m_lqcache. Always as big as MAX (rows, cols).

Definition at line 141 of file ImageComparison.hh.

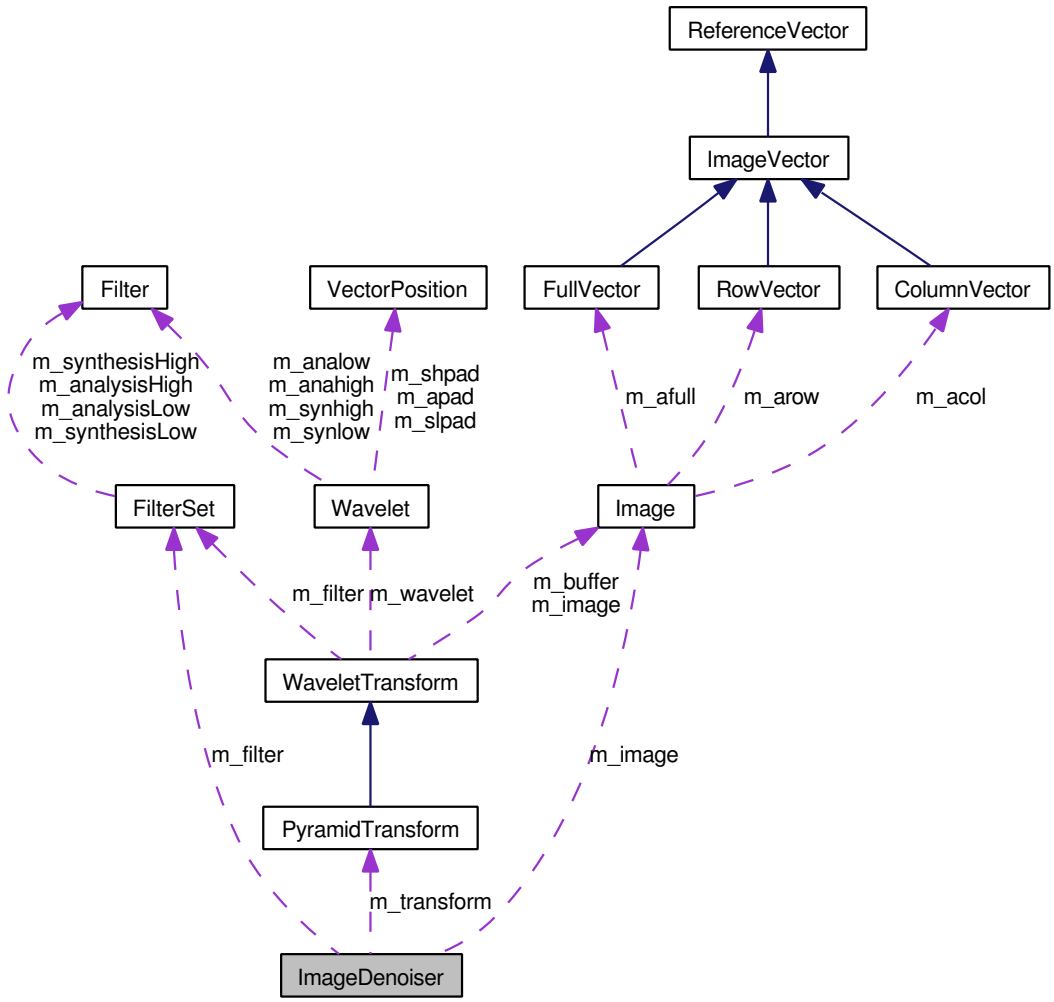
The documentation for this class was generated from the following file:

- ImageComparison.hh

8.30 ImageDenoiser Class Reference

`#include <ImageDenoiser.hh>`

Collaboration diagram for ImageDenoiser:



Public Member Functions

- ImageDenoiser (Image &img, unsigned areas, int fromSub, int toSub=1, double alpha=1.0, FilterSet &filter=Daub4, int isSingleSignificant=0, int replacementFunction=0)
- virtual ~ImageDenoiser (void)
- virtual void denoise (void)

Protected Member Functions

- coeff threshold (int subband, area channel)
- virtual void processFrom (PyramidTree &tree, coeff *thresholds, area channel, int level)
- virtual bool isCoeffSingleSignificant (CoeffInformation &c, area channel, coeff threshold, bool &stop)
- virtual bool isSingleSignificantInRegion (CoeffInformation &c, area channel, coeff threshold, bool &stop)
- virtual bool isSingleSignificantInChannel (CoeffInformation &c, area channel, coeff threshold, bool &stop)
- virtual void replaceSimple (int y, int x, area channel, coeff threshold)
- virtual void replaceChannel (int y, int x, area channel, coeff threshold)

Protected Attributes

- Image * m_image
- PyramidTransform * m_transform
- FilterSet * m_filter
- int m_from
- int m_to
- unsigned m_areas
- double m_alpha
- int m_significanceFunction
- int m_replacementFunction

8.30.1 Detailed Description

A simple DWT-based image denoiser. It uses the same tree structure as in Shapiro's Zerotree compression algorithm. Denoising means reducing the power of significant coefficients in finer subbands that have insignificant parents in the coarser subbands.

Definition at line 39 of file ImageDenoiser.hh.

8.30.2 Constructor & Destructor Documentation

8.30.2.1 ImageDenoiser::ImageDenoiser (Image & *img*, unsigned *areas*, int *fromSub*, int *toSub* = 1, double *alpha* = 1.0, FilterSet & *filter* = Daub4, int *isSingleSignificant* = 0, int *replacementFunction* = 0)

Constructor, initializes internal structures, sets up preferences.

Parameters:

img a reference to the image to operate on.

areas the areas to operate on. Use a binary OR to combine more than one area (see macros above).

fromSub the subband to start building the trees

toSub the target subband where the tree terminates

alpha the scaling factor to the standard deviation for determining the significance threshold

filter the filterset for decomposition

isSingleSignificant the kind of function used to determine significance, 0: single values, 1: 3x3 regions, 2: channels in 3x3 environments

replacementFunction the kind of function used to replace coefficient we consider as noise, 0: +-0.9∗threshold, 1: more complicated formula depending on channel (see below)

Exceptions:

invalid_argument if the areas argument is zero or not at least two subbands are selected or an invalid significance function is selected

8.30.2.2 virtual ImageDenoiser::∼ImageDenoiser (void) [virtual]

Destructor, clean up.

8.30.3 Member Function Documentation

8.30.3.1 virtual void ImageDenoiser::denoise (void) [virtual]

Apply the denoising algorithm to the image.

Exceptions:

invalid_argument if some error occurs during decomposition or reconstruction of the image

8.30.3.2 coeff ImageDenoiser::threshold (int *subband*, area *channel*) [protected]

Calculate the threshold for a given area/subband.

Parameters:

channel the area

subband the subband

Returns:

the threshold

8.30.3.3 virtual void ImageDenoiser::processFrom (PyramidTree & *tree*, coeff * *thresholds*, area *channel*, int *level*) [protected, virtual]

Process one tree or subtree by applying the thresholds

Parameters:

tree the tree structure to process

thresholds an array holding depth thresholds

channel which area inside a subband

level the current position's decomposition level

8.30.3.4 virtual bool ImageDenoiser::isCoeffSingleSignificant (CoeffInformation & *c*, area *channel*, coeff *threshold*, bool & *stop*) [protected, virtual]

Return true if the coefficient's value is significant with respect to the threshold.

Parameters:

c the coefficient

channel which area inside a subband

threshold the threshold

stop set to true if we found an edge and want the processing of this tree to stop (always false here)

Returns:

true if the coefficient is significant

8.30.3.5 virtual bool ImageDenoiser::isSingleSignificantInRegion (CoeffInformation & *c*, area *channel*, coeff *threshold*, bool & *stop*) [protected, virtual]

Return true if the coefficient is significant with respect to the threshold but the surrounding square region is not.

Parameters:

c the coefficient

channel which area inside a subband

threshold the threshold

stop set to true if we found an edge and want the processing of this tree to stop

Returns:

true if the coefficient's region is significant

8.30.3.6 virtual bool ImageDenoiser::isSingleSignificantInChannel (CoeffInformation & *c*, area *channel*, coeff *threshold*, bool & *stop*) [protected, virtual]

Return true if the coefficient is significant with respect to the threshold but the surrounding channel-dependent region is not.

Parameters:

c the coefficient

channel which area inside a subband

threshold the threshold

stop set to true if we found an edge and want the processing of this tree to stop

Returns:

true if the coefficient's region is significant

8.30.3.7 virtual void ImageDenoiser::replaceSimple (int *y*, int *x*, area *channel*, coeff *threshold*) [protected, virtual]

Replace a coefficient that we think is noise. This method simply replaces it by the threshold multiplied by the original value's sign and 0.99.

Parameters:

y the row in the image

x the column in the image

channel which area inside a subband

threshold the threshold

8.30.3.8 virtual void ImageDenoiser::replaceChannel (int *y*, int *x*, area *channel*, coeff *threshold*) [protected, virtual]

Replace a coefficient that we think is noise. This method uses the formula found in "Image Denoising Using a Local Gaussian Scale Mixture Model in the Wavelet Domain" by Strela/Portilla/Simoncelli: I_xy = sqrt (sum_n(I_xnyn^2)/n), however we only count vertical, horizontal or diagonal values depending on which channel a value is in.

Parameters:

y the row in the image

x the column in the image

channel which area inside a subband

threshold the threshold

8.30.4 Member Data Documentation

8.30.4.1 Image* ImageDenoiser::m_image [protected]

A reference to the image to process.

Definition at line 139 of file ImageDenoiser.hh.

8.30.4.2 PyramidTransform* ImageDenoiser::m_transform [protected]

The wavelet transform to be used.

Definition at line 141 of file ImageDenoiser.hh.

8.30.4.3 FilterSet* ImageDenoiser::m_filter [protected]

The filterset used for decomposition

Definition at line 143 of file ImageDenoiser.hh.

8.30.4.4 int ImageDenoiser::m_from [protected]

The subband to start building the trees

Definition at line 145 of file ImageDenoiser.hh.

8.30.4.5 int ImageDenoiser::m_to [protected]

The target subband where the tree terminates

Definition at line 147 of file ImageDenoiser.hh.

8.30.4.6 unsigned ImageDenoiser::m_areas [protected]

The areas to operate on. Use a binary OR to combine more than one area (see macros above).

Definition at line 150 of file ImageDenoiser.hh.

8.30.4.7 double ImageDenoiser::m_alpha [protected]

The scaling factor to the standard deviation for determining the significance threshold

Definition at line 153 of file ImageDenoiser.hh.

8.30.4.8 int ImageDenoiser::m_significanceFunction [protected]

The significance function used.

Definition at line 155 of file ImageDenoiser.hh.

8.30.4.9 int ImageDenoiser::m_replacementFunction [protected]

The replacement function used.

Definition at line 157 of file ImageDenoiser.hh.

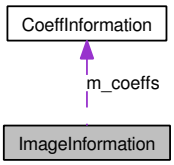
The documentation for this class was generated from the following file:

- ImageDenoiser.hh

8.31 ImageInformation Class Reference

`#include <ImageInformation.hh>`

Collaboration diagram for ImageInformation:



Public Member Functions

- ImageInformation (int size=0)
- ImageInformation (Image &img)
- ~ImageInformation (void)
- int size (void) const
- ImageInformation * head (int n)
- ImageInformation * tail (int n)
- ImageInformation * append (ImageInformation &ii)
- ImageInformation * clone (void)
- void shrink (int newsize, bool leavehead=true)
- bool equals (ImageInformation &ii)
- CoeffInformation & at (int pos)
- void to (CoeffInformation &c, int pos)
- void to (const CoeffInformation &c, int pos)
- void resize (int size)
- void swap (int pos1, int pos2)
- void gensort (cipredicate isless)
- void ssort (void)
- void asort (void)
- void psort (void)
- void yxsort (void)
- void xysort (void)
- void isort (void)
- void shuffle (void)
- bool ssorted (void) const
- bool asorted (void) const
- bool psorted (void) const
- bool isorted (void) const
- bool isIn (int abs) const
- bool isIn (int y, int x) const
- int locate (int abs) const
- int locate (int y, int x) const

- coeff amax (void)
- coeff smax (void)
- coeff amin (void)
- coeff smin (void)
- void dump (const char *delim=" ", const char *file="") const
- coeff aaverage (void) const
- coeff saverage (void) const
- double sqvariance (void) const
- double variance (bool abs=false) const
- double sdeviation (bool abs=false) const

Protected Member Functions

- void copyCoeffs (CoeffInformation *coeffs, int size, bool head)
- ImageInformation * subimage (int size, bool head)
- int randint (int from, int to)
- void quicksort (int from, int to, cipredicate isless)

Protected Attributes

- CoeffInformation * m_coeffs
- int m_size

8.31.1 Detailed Description

Image information. The information consists coefficient information nodes, so the original image can be reconstructed from this. Tool methods for sorting, shuffleing etc. are used for watermark.

Definition at line 35 of file ImageInformation.hh.

8.31.2 Constructor & Destructor Documentation

8.31.2.1 ImageInformation::ImageInformation (int *size* = 0)

Constructor. Initializes a number of empty components.

Parameters:

size the number of components

8.31.2.2 ImageInformation::ImageInformation (Image & *img*)

Constructor. Gets information from an image.

Parameters:

img the source image

8.31.2.3 ImageInformation::~ImageInformation (void)

Destructor. Frees allocated objects.

8.31.3 Member Function Documentation

8.31.3.1 int ImageInformation::size (void) const [inline]

Get the number of stored coeffs.

Returns:

the number of stored coeffs

Definition at line 51 of file ImageInformation.hh.

8.31.3.2 ImageInformation* ImageInformation::head (int *n*)

Get the first n coeffs.

Parameters:

n the number of coeffs

Returns:

a new info object containing the coeffs

8.31.3.3 ImageInformation* ImageInformation::tail (int *n*)

Get the last n coeffs.

Parameters:

n the number of coeffs

Returns:

a new info object containing the coeffs

8.31.3.4 ImageInformation* ImageInformation::append (ImageInformation & *ii*)

Append another ImageInformation object.

Parameters:

ii the other ImageInformation object.

Returns:

a new info object containing the coeffs

8.31.3.5 ImageInformation* ImageInformation::clone (void) [inline]

Get a copy of this.

Returns:

a new info object containing a copy of this

Definition at line 73 of file ImageInformation.hh.

8.31.3.6 void ImageInformation::shrink (int *newsize*, bool *leavehead* = true)

Shrink (discard some of the coeffs).

Parameters:

newsize the new number of coeffs, if greater than the actual size, the operation has no effect

leavehead if {*true*} discard from behind, else discard from the top

8.31.3.7 bool ImageInformation::equals (ImageInformation & *ii*)

Equality test.

Parameters:

ii other info object

Returns:

if number of coeffs, order and the coeffs themselves are equal: {*true*}, else {*false*}

8.31.3.8 CoeffInformation& ImageInformation::at (int *pos*) [inline]

Return a coeff info.

Parameters:

pos the info's position in this object (array index)

Returns:

a reference to the coeff info node

Definition at line 103 of file ImageInformation.hh.

8.31.3.9 void ImageInformation::to (CoeffInformation & *c*, int *pos*) [inline]

Set a position from a CoeffInformation object.

Parameters:

c the CoeffInformation object

pos the info's position in this object (array index)

Definition at line 109 of file ImageInformation.hh.

8.31.3.10 void ImageInformation::to (const CoeffInformation & *c*, int *pos*) [inline]

Set a position from a CoeffInformation object.

Parameters:

c the CoeffInformation object

pos the info's position in this object (array index)

Definition at line 115 of file ImageInformation.hh.

8.31.3.11 void ImageInformation::resize (int *size*)

Resize the object preserving as many old values as possible.

Parameters:

size the new size

8.31.3.12 void ImageInformation::swap (int *pos1*, int *pos2*)

Swap two coefficients.

Parameters:

pos1 the first object's position

pos2 the second object's position

8.31.3.13 void ImageInformation::gensort (cipredicate *isless*)

Sort the coeffs according to their values.

Parameters:

isless the comparison function to use.

8.31.3.14 void ImageInformation::ssort (void)

Sort the coeffs according to their values.

8.31.3.15 void ImageInformation::asort (void)

Sort the coeffs according to their absolute values.

8.31.3.16 void ImageInformation::psort (void)

Sort the coeffs according to their absolute positions in the image.

8.31.3.17 void ImageInformation::yxsort (void)

Sort the coeffs according to their rows/cols positions in the image.

8.31.3.18 void ImageInformation::xysort (void)

Sort the coeffs according to their cols/rows positions in the image.

8.31.3.19 void ImageInformation::isort (void)

Sort the coeffs according to their IDs.

8.31.3.20 void ImageInformation::shuffle (void)

Mix coeffs using pseudo-random numbers.

8.31.3.21 bool ImageInformation::ssorted (void) const

Are the coeffs sorted according to their values?

Returns:

if sorted according to their values: {*true*}, else {*false*}

8.31.3.22 bool ImageInformation::asorted (void) const

Are the coeffs sorted according to their absolute values?

Returns:

if sorted according to their values: {*true*}, else {*false*}

8.31.3.23 bool ImageInformation::psorted (void) const

Are the coeffs sorted according to their position in their image?

Returns:

if sorted according to their positions: {*true*}, else {*false*}

8.31.3.24 bool ImageInformation::isorted (void) const

Are the coeffs sorted according to their IDs?

Returns:

if sorted according to their IDs: {*true*}, else {*false*}

8.31.3.25 bool ImageInformation::isIn (int *abs*) const [inline]

Returns true if a position is in this ImageInformation object.

Parameters:

abs the absolute position in the image

Returns:

true if the position was found

Definition at line 163 of file ImageInformation.hh.

8.31.3.26 bool ImageInformation::isIn (int *y*, int *x*) const [inline]

Returns true if a position is in this ImageInformation object.

Parameters:

y the row in the image

x the col in the image

Returns:

true if the position was found

Definition at line 168 of file ImageInformation.hh.

8.31.3.27 int ImageInformation::locate (int *abs*) const

Returns true a position's index if it is is in this ImageInformation object.

Parameters:

abs the absolute position in the image

Returns:

the index if found, else -1

8.31.3.28 int ImageInformation::locate (int *y*, int *x*) const

Returns true a position's index if it is is in this ImageInformation object.

Parameters:

y the row in the image

x the col in the image

Returns:

the index if found, else -1

8.31.3.29 coeff ImageInformation::amax (void)

Get the maximum signed value from the selected coefficients.

Returns:

the maximum value

8.31.3.30 coeff ImageInformation::smax (void)

Get the maximum absolute value from the selected coefficients.

Returns:

the maximum value

8.31.3.31 coeff ImageInformation::amin (void)

Get the minimum absolute value from the selected coefficients.

Returns:

the minimum value

8.31.3.32 coeff ImageInformation::smin (void)

Get the minimum signed value from the selected coefficients.

Returns:

the minimum value

8.31.3.33 void ImageInformation::dump (const char * *delim* = " ", const char * *file* = "") const

Write the contents to stdout or a file.

Exceptions:

ios_base::failure if the file could not be opened for writing.

Parameters:

delim what to print between two entries

file the name of the file (empty string for stdout)

8.31.3.34 coeff ImageInformation::aaverage (void) const

Return the average absolute greyscale value.

Returns:

the average absolute greyscale value

8.31.3.35 coeff ImageInformation::saverage (void) const

Return the average (signed) greyscale value.

Returns:

the average greyscale value

8.31.3.36 double ImageInformation::sqvariance (void) const

Return the square variance.

Returns:

the square variance

8.31.3.37 double ImageInformation::variance (bool *abs* = `false`) const

Return the variance.

Parameters:

abs true if signs are discarded.

Returns:

the variance

8.31.3.38 double ImageInformation::sdeviation (bool *abs* = `false`) const

Returns the standard deviation.

Parameters:

abs true if signs are discarded.

Returns:

the standard deviation.

8.31.3.39 void ImageInformation::copyCoeffs (CoeffInformation * *coeffs*, int *size*, bool *head*) [protected]

Obtain a subset of the coeffs.

Parameters:

coeffs the address of the CoeffInformation array to write to

size the new array's size

head if {*true}* the subset will be taken from the top, else from the bottom

8.31.3.40 ImageInformation* ImageInformation::subimage (int *size*, bool *head*) [protected]

Obtain a new info object containging a subset of the coeffs.

Parameters:

size the number of coeffs to be stored

head if {*true*} the subset will be taken from the top, else from the bottom

Returns:

the new info object

8.31.3.41 int ImageInformation::randint (int *from*, int *to*) [protected]

Obtain positive pseudo-random numbers.

Parameters:

from the minimum number

to one more than the maximum number

Returns:

the pseudo-random number

8.31.3.42 void ImageInformation::quicksort (int *from*, int *to*, cipredicate *isless*) [protected]

Sort the coefficients. A Quicksort algorithm according to Jon Bentley.

Parameters:

from the start index

to the end index (included in sort)

isless the the comparison function, {*true*} when the left arg is less than the right one

8.31.4 Member Data Documentation

8.31.4.1 CoeffInformation* ImageInformation::m_coeffs [protected]

An array containing the coefficient nodes.

Definition at line 238 of file ImageInformation.hh.

8.31.4.2 int ImageInformation::m_size [protected]

The number of coeffs.

Definition at line 240 of file ImageInformation.hh.

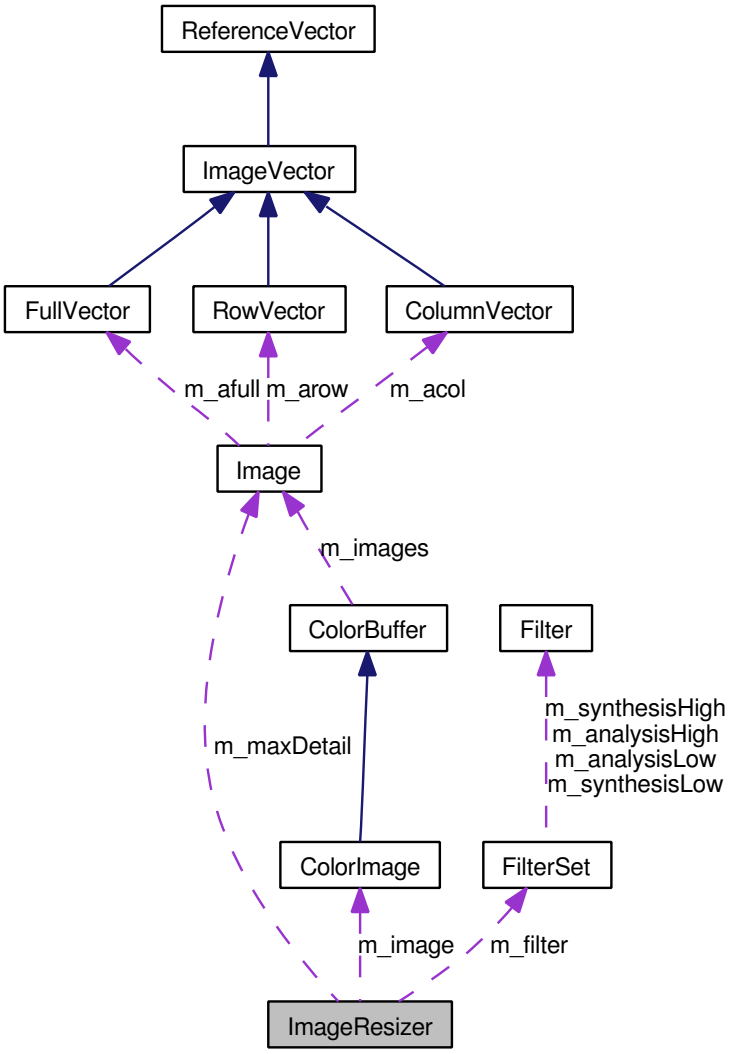
The documentation for this class was generated from the following file:

- ImageInformation.hh

8.32 ImageResizer Class Reference

`#include <ImageResizer.hh>`

Collaboration diagram for ImageResizer:



Public Member Functions

- ImageResizer (const ColorImage &img, FilterSet &flt, int *fill=NULL, coeff threshold=0.001, bool optimizeImage=false, int scalingStrategy=0)
- ~ImageResizer (void)
- ColorImage * resize (int rows, int cols, int steps=1)
- ColorImage * redimension (int rows, int cols, int steps=1)
- double getInnerAvgPerSize (void) const

- double getInnerSDeviation (void) const
- int getInnerRegionSize (void) const
- double threshold (void) const
- void threshold (double threshold)
- void optimizeImage (bool optimizeImage)
- bool optimizeImage (void)

Protected Types

- enum { CROP_COLS, CROP_ROWS, CROP_BOTH_OR_NONE, CROP_AUTOMATICALLY }

Protected Member Functions

- void calcDimensions (int rows, int cols)
- int fixDimensions (int rows, int cols)
- void calcOptimization (void)
- void calcOuterStats (coeff &avg, coeff &variance, int firstTop, int firstLeft, int firstBottom, int firstRight, int secondTop, int secondLeft, int secondBottom, int secondRight)
- void calcInnerStats (coeff &avg, coeff &sDev, int top, int left, int bottom, int right)
- void calcStats (int nRows, int nCols, coeff &outerAvg, coeff &outerSDev, coeff &innerAvg, coeff &innerSDev)
- void genMaxDetail (int steps)
- ColorImage * getUnscaledImage (int rows, int cols, int discardRows, int discardCols)
- ColorImage * getScaledImage (int rows, int cols, int discardRows, int discardCols)
- ColorImage * getCroppedImage (int discardRows, int discardCols)
- ColorImage * doResize (int rows, int cols, int steps, bool redimensionOnly)

Protected Attributes

- const ColorImage & m_image
- Image * m_maxDetail
- FilterSet & m_filter
- float m_qRows
- float m_qCols
- int * m_fill
- int m_scalingStrategy
- double m_innerAvgPerSize
- double m_innerSDeviation
- int m_innerRegionSize
- coeff m_threshold
- bool m_optimizeImage

- bool m_optimizationCalculated
- int m_cropMaxRows
- int m_cropMaxCols
- double m_rowsMapping
- double m_colsMapping
- int m_steps
- enum ImageResizer:: { ... } m_cropWhat

8.32.1 Detailed Description

A class for advanced resizing of images, i.e. wavelet-based techniques can be used to intelligently crop images to fit

This class will only discard background pixels if the image is shrunk.

Definition at line 26 of file ImageResizer.hh.

8.32.2 Member Enumeration Documentation

8.32.2.1 anonymous enum [protected]

Enumerator:

CROP_COLS

CROP_ROWS

CROP_BOTH_OR_NONE

CROP_AUTOMATICALLY

Definition at line 296 of file ImageResizer.hh.

8.32.3 Constructor & Destructor Documentation

8.32.3.1 ImageResizer::ImageResizer (const ColorImage & *img*, FilterSet & *flt*, int * *fill* = NULL, coeff *threshold* = 0.001, bool *optimizeImage* = false, int *scalingStrategy* = 0)

Constructor, loads an image, sets Wavelet to be used.

Parameters:

img a reference to the image

flt a reference to the filterset

fill if not NULL it must point to an array of as many values as color channels, so that each of the image's channels has its own fill greyscale value for the remaining space (else the a smaller image size will be chosen if the aspect ratio does not match)

optimizeImage if set to true, we will try to locate the actual contents regardless of whether the target dimensions fit or not

threshold the threshold for the aggressiveness of the cropping (< 1), the higher the more aggressive, default is 0.001

scalingStrategy the scaling strategy (0: bilinear interpolation, 1: average, 2: nearest neighbour).

Returns:

a new rescaled image

8.32.3.2 ImageResizer::~ImageResizer (void)

Destructor. Cleans up if necessary.

8.32.4 Member Function Documentation

8.32.4.1 ColorImage* ImageResizer::resize (int *rows*, int *cols*, int *steps* = 1)

Create a resized image's of x/y dimensions. The result image will be cropped to optimally fit in the new dimensions. The maximal percentage of cropping as well as the thresholds for guessing insignificant areas can be set. If both dimensions are set to 0 and the optimization feature is set, we get an auto-crop (i.e. the size is chosen from the image's contents) The new dimensions must all be greater than zero.

Exceptions:

invalid_argument one or both dimensions are either negative or zero

Parameters:

rows the new number of rows

cols the new number of cols

steps the number of steps to decompose before checking for features

Returns:

the new resized image

8.32.4.2 ColorImage* ImageResizer::redimension (int *rows*, int *cols*, int *steps* = 1)

Prepare an image for resizing to x/y dimensions. The result image will be cropped to optimally fit in the new dimensions. The maximal percentage of cropping as well as the thresholds for guessing insignificant areas can be set. This operation skips the actual scaling, so that it can be performed by someone else. The new dimensions must all be greater than zero.

Exceptions:

invalid_argument one or both dimensions are either negative or zero

Parameters:

rows the new number of rows

cols the new number of cols

steps the number of steps to decompose before checking for features

Returns:

the new redimensioned image

8.32.4.3 double ImageResizer::getInnerAvgPerSize (void) const [inline]

Return the inner (absolute) average per size (used for quality measurements)

Returns:

the inner average per size

Definition at line 95 of file ImageResizer.hh.

References m_innerAvgPerSize.

8.32.4.4 double ImageResizer::getInnerSDeviation (void) const [inline]

Return the inner standard deviation (used for quality measurements)

Returns:

the inner standard deviation

Definition at line 100 of file ImageResizer.hh.

References m_innerSDeviation.

8.32.4.5 int ImageResizer::getInnerRegionSize (void) const [inline]

Return the size of the inner region used for the above calculation

Returns:

the size of the inner region used for the above calculation

Definition at line 105 of file ImageResizer.hh.

References m_innerRegionSize.

8.32.4.6 double ImageResizer::threshold (void) const [inline]

Return the threshold for cropping-while-resizing

Returns:

the threshold for cropping-while-resizing

Definition at line 111 of file ImageResizer.hh.

References m_threshold.

8.32.4.7 void ImageResizer::threshold (double *threshold*) [inline]

Set the threshold for cropping-while-resizing

Parameters:

threshold the threshold for cropping-while-resizing

Definition at line 116 of file ImageResizer.hh.

References m_threshold.

8.32.4.8 void ImageResizer::optimizeImage (bool *optimizeImage*) [inline]

Set the optimize-image-setting

Parameters:

optimize true if optimization is active

Definition at line 122 of file ImageResizer.hh.

References m_optimizeImage.

8.32.4.9 bool ImageResizer::optimizeImage (void) [inline]

Return the optimize-image-setting

Returns:

true if optimization is active

Definition at line 127 of file ImageResizer.hh.

References m_optimizeImage.

8.32.4.10 void ImageResizer::calcDimensions (int *rows*, int *cols*) [protected]

Calculate the image's new dimensions

Parameters:

rows the number of rows

cols the number of cols

8.32.4.11 int ImageResizer::fixDimensions (int *rows*, int *cols*) [protected]

If the target dimensions do not match the source dimensions, determine how many rows or cols respectively can be discarded to further optimize the image.

Parameters:

rows the target rows

cols the target cols

Returns:

the number of rows / cols which can be discarded in the original image

8.32.4.12 void ImageResizer::calcOptimization (void) [protected]

Determine how many rows and cols can be discarded regardless of the target dimensions. This is calculated once within resize(), so that if more than one target size is required by repeated resizing operations little or no further effort is necessary after the first operation.

8.32.4.13 void ImageResizer::calcOuterStats (coeff & *avg*, coeff & *variance*, int *firstTop*, int *firstLeft*, int *firstBottom*, int *firstRight*, int *secondTop*, int *secondLeft*, int *secondBottom*, int *secondRight*) [protected]

Calculate the average and variance for an outer region of the (internal) maxDetail image. This is needed to see whether we can discard this outer region or not.

Parameters:

avg the calculated average

variance the calculated variance

firstTop the first top of the frame

firstLeft the first left of the frame

firstBottom the first bottom of the frame

firstRight the first right of the frame

secondTop the second top of the frame

secondLeft the second left of the frame

secondBottom the second bottom of the frame

secondRight the second right of the frame

8.32.4.14 void ImageResizer::calcInnerStats (coeff & *avg*, coeff & *sDev*, int *top*, int *left*, int *bottom*, int *right*) [protected]

Calculate the average and variance for an inner region of the (internal) maxDetail image. This is needed to see whether we can discard the region around it or not.

Parameters:

avg the calculated average

variance the calculated variance

top the top of the inner region

left the left of the inner region

bottom the bottom of the inner region

right the right of the inner region

8.32.4.15 void ImageResizer::calcStats (int *nRows*, int *nCols*, coeff & *outerAvg*, coeff & *outerSDev*, coeff & *innerAvg*, coeff & *innerSDev*) [protected]

Calculate average and standard deviation for the regions either top and bottom or left and right (depending on the value of m_cropWhat) and the region between for a given number of rows/columns symetrically from the boundaries to the center of the image.

Parameters:

nRows the number of rows over which to calculate

nCols the number of rows over which to calculate

outerAvg (out parameter) is where the calculated average for the regions to crop is stored

outerSDev (out parameter) is where the calculated standard deviation for the regions to crop is stored

innerAvg (out parameter) is where the calculated average for the region between is stored

innerSDev (out parameter) is where the calculated standard deviation for the region between is stored

8.32.4.16 void ImageResizer::genMaxDetail (int *steps*) [protected]

Calculate the maxDetail image for the statistics

Parameters:

steps the number of decomposition steps

8.32.4.17 ColorImage* ImageResizer::getUnscaledImage (int *rows*, int *cols*, int *discardRows*, int *discardCols*) [protected]

Create the final unscaled result image using the previously calculated number of vectors to discard.

Parameters:

rows the number of rows

cols the number of cols

discardRows the number of rows to discard on each side (i.e. we actually discard 2 * discardRows overall); if less than or equal to zero, nothing will be discarded

discardCols the number of rows to discard on each side (i.e. we actually discard 2 * discardCols overall); if less than or equal to zero, nothing will be discarded

Returns:

the result image

8.32.4.18 ColorImage* ImageResizer::getScaledImage (int *rows*, int *cols*, int *discardRows*, int *discardCols*) [protected]

Create the final scaled result image using the previously calculated number of vectors to discard.

Parameters:

rows the number of rows

cols the number of cols

discardRows the number of rows to discard on each side (i.e. we actually discard 2 * discardRows overall); if less than or equal to zero, nothing will be discarded

discardCols the number of rows to discard on each side (i.e. we actually discard 2 * discardCols overall); if less than or equal to zero, nothing will be discarded

Returns:

the result image

8.32.4.19 ColorImage* ImageResizer::getCroppedImage (int *discardRows*, int *discardCols*) [protected]

This operation applies the cropping using the previously calculated numbers of rows or cols which can be safely discarded.

Parameters:

discardRows the number of rows to discard on each side (i.e. we actually discard 2 * discardRows overall); if less than or equal to zero, nothing will be discarded

discardCols the number of rows to discard on each side (i.e. we actually discard 2 * discardCols overall); if less than or equal to zero, nothing will be discarded

Returns:

the cropped image or a copy of the original if nothing was cropped

8.32.4.20 ColorImage* ImageResizer::doResize (int *rows*, int *cols*, int *steps*, bool *redimensionOnly*) [protected]

Create a resized image's of x/y dimensions. The result image will be cropped to optimally fit in the new dimensions. The maximal percentage of cropping as well as the thresholds for guessing insignificant areas can be set. If both dimensions are set to 0 and the optimization feature is set, we get an auto-crop (i.e. the size is chosen from the image's contents) The new dimensions must all be greater than zero.

Exceptions:

invalid_argument one or both dimensions are either negative or zero

Parameters:

rows the new number of rows

cols the new number of cols

steps the number of steps to decompose before checking for features

redimensionOnly if true, the image will be prepared for scaling, i.e. the dimensions are fixed filling performed etc., so that it can be scaled by some other application later.

Returns:

the new resized or prepared image

8.32.5 Member Data Documentation

8.32.5.1 const ColorImage& ImageResizer::m_image [protected]

Definition at line 278 of file ImageResizer.hh.

8.32.5.2 Image* ImageResizer::m_maxDetail [protected]

Definition at line 279 of file ImageResizer.hh.

8.32.5.3 FilterSet& ImageResizer::m_filter [protected]

Definition at line 280 of file ImageResizer.hh.

8.32.5.4 float ImageResizer::m_qRows [protected]

Definition at line 281 of file ImageResizer.hh.

8.32.5.5 float ImageResizer::m_qCols [protected]

Definition at line 282 of file ImageResizer.hh.

8.32.5.6 int* ImageResizer::m_fill [protected]

Definition at line 283 of file ImageResizer.hh.

8.32.5.7 int ImageResizer::m_scalingStrategy [protected]

Definition at line 284 of file ImageResizer.hh.

8.32.5.8 double ImageResizer::m_innerAvgPerSize [protected]

Definition at line 285 of file ImageResizer.hh.

Referenced by getInnerAvgPerSize().

8.32.5.9 double ImageResizer::m_innerSDeviation [protected]

Definition at line 286 of file ImageResizer.hh.

Referenced by getInnerSDeviation().

8.32.5.10 int ImageResizer::m_innerRegionSize [protected]

Definition at line 287 of file ImageResizer.hh.

Referenced by getInnerRegionSize().

8.32.5.11 coeff ImageResizer::m_threshold [protected]

Definition at line 288 of file ImageResizer.hh.

Referenced by threshold().

8.32.5.12 bool ImageResizer::m_optimizeImage [protected]

Definition at line 289 of file ImageResizer.hh.

Referenced by optimizeImage().

8.32.5.13 bool ImageResizer::m_optimizationCalculated [protected]

Definition at line 290 of file ImageResizer.hh.

8.32.5.14 int ImageResizer::m_cropMaxRows [protected]

Definition at line 291 of file ImageResizer.hh.

8.32.5.15 int ImageResizer::m_cropMaxCols [protected]

Definition at line 292 of file ImageResizer.hh.

8.32.5.16 double ImageResizer::m_rowsMapping [protected]

Definition at line 293 of file ImageResizer.hh.

8.32.5.17 double ImageResizer::m_colsMapping [protected]

Definition at line 294 of file ImageResizer.hh.

8.32.5.18 int ImageResizer::m_steps [protected]

Definition at line 295 of file ImageResizer.hh.

8.32.5.19 enum { ... } ImageResizer::m_cropWhat [protected]

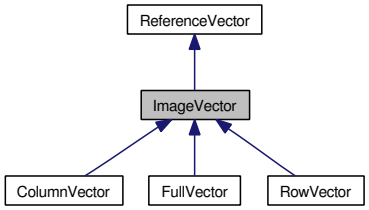
The documentation for this class was generated from the following file:

- ImageResizer.hh

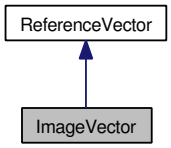
8.33 ImageVector Class Reference

`#include <ImageVector.hh>`

Inheritance diagram for ImageVector:



Collaboration diagram for ImageVector:



Public Member Functions

- ImageVector (ImageArray< coeff > *ar)
- virtual ~ImageVector (void)
- virtual bool sanity (void)
- virtual void update (void)
- virtual void go (int root)
- virtual coeff at (int pos)=0
- virtual void to (int pos, coeff val)=0
- virtual int size (void)=0
- coeff weight (void)

Protected Attributes

- ImageArray< coeff > * m_array
- int m_xsize
- int m_ysize

8.33.1 Detailed Description

Pseudo-vector (abstract) for use with two-dimensional objects, like images. Provide a two-dimensional array's rows and columns as vectors with array addressing.

Definition at line 26 of file ImageVector.hh.

8.33.2 Constructor & Destructor Documentation

8.33.2.1 ImageVector::ImageVector (ImageArray< coeff > * *ar*)

Constructor. Sets the array reference.

8.33.2.2 virtual ImageVector::~ImageVector (void) [inline, virtual]

Destructor. Does nothing.

Definition at line 34 of file ImageVector.hh.

8.33.3 Member Function Documentation

8.33.3.1 virtual bool ImageVector::sanity (void) [virtual]

Checks integrity. If the array's dimensions have been changed (resize operation) {*false*} will be returned.

Implements ReferenceVector.

8.33.3.2 virtual void ImageVector::update (void) [virtual]

Updates the vector's settings. This is necessary each time after the array has been resized.

Implements ReferenceVector.

Reimplemented in ColumnVector, FullVector, and RowVector.

8.33.3.3 virtual void ImageVector::go (int *root*) [virtual]

Set new root. Depending of what concrete instance is the current row or col will be set.

Exceptions:

invalid_argument a negative value was given for the new root

Parameters:

root the new root

Implements ReferenceVector.

Reimplemented in ColumnVector, FullVector, and RowVector.

8.33.3.4 virtual coeff ImageVector::at (int *pos*) [pure virtual]

Return a value (abstract). The value is taken from the vector's position {*pos*}.

Parameters:

pos the position

Returns:

the value at that position

Implements ReferenceVector.

Implemented in ColumnVector, FullVector, and RowVector.

8.33.3.5 virtual void ImageVector::to (int *pos*, coeff *val*) [pure virtual]

Assign a value (abstract). A new value {*val*} is assigned to the vector's position {*pos*}.

Parameters:

pos the position

val the new value

Implements ReferenceVector.

Implemented in ColumnVector, FullVector, and RowVector.

8.33.3.6 virtual int ImageVector::size (void) [pure virtual]

Return the vector's size (abstract). Depending on the concrete instance we will get the associated array's number of rows or cols.

Returns:

the vector's size

Implements ReferenceVector.

Implemented in ColumnVector, FullVector, and RowVector.

8.33.3.7 coeff ImageVector::weight (void)

Return the sum of the vector's absolute values

Returns:

the sum of the vector's absolute values

8.33.4 Member Data Documentation

8.33.4.1 ImageArray<coeff>* ImageVector::m_array [protected]

A reference to encapsuled array. Rows and cols will be taken from this array.

Definition at line 80 of file ImageVector.hh.

8.33.4.2 int ImageVector::m_xsize [protected]

The encapsulated array's cols.

Definition at line 82 of file ImageVector.hh.

Referenced by FullVector::FullVector().

8.33.4.3 int ImageVector::m_ysize [protected]

The encapsulated array's rows .

Definition at line 84 of file ImageVector.hh.

Referenced by FullVector::FullVector().

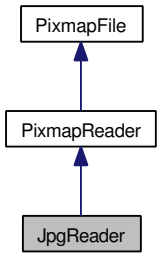
The documentation for this class was generated from the following file:

- ImageVector.hh

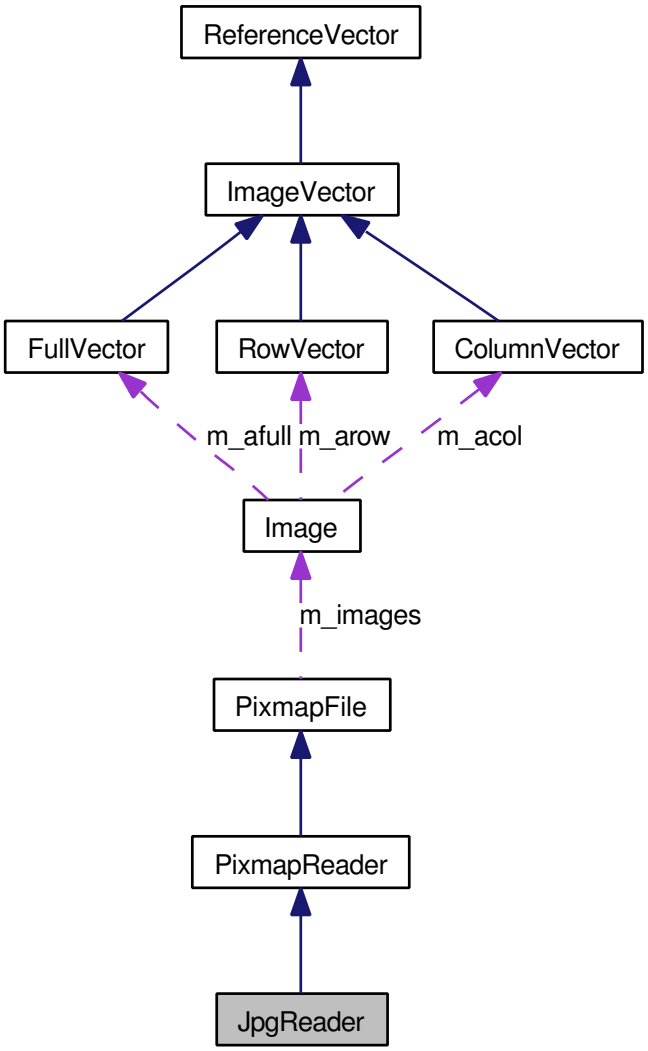
8.34 JpgReader Class Reference

`#include <JpgReader.hh>`

Inheritance diagram for JpgReader:



Collaboration diagram for JpgReader:



Public Member Functions

- JpgReader (char const *name, Image *images[ ])

Protected Member Functions

- virtual int readfmt (void)

8.34.1 Detailed Description

A JPG file reader. Only JPGs version 6 are supported.

BUGS: Only color depths of 1 and 3 are supported.

Definition at line 23 of file JpgReader.hh.

8.34.2 Constructor & Destructor Documentation

8.34.2.1 JpgReader::JpgReader (char const * *name*, Image * *images*[ ]) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference.

Parameters:

name the file name

images the array of greyscale images for the values

Definition at line 33 of file JpgReader.hh.

8.34.3 Member Function Documentation

8.34.3.1 virtual int JpgReader::readfmt (void) [protected, virtual]

Read the JPG file format. This does the actual work of reading and parsing the image file. It gets called by the {*read* ()} method.

Returns:

0 if successful, -1 on read error, -2 on file format error.

Implements PixmapReader.

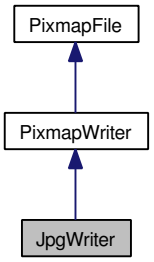
The documentation for this class was generated from the following file:

- JpgReader.hh

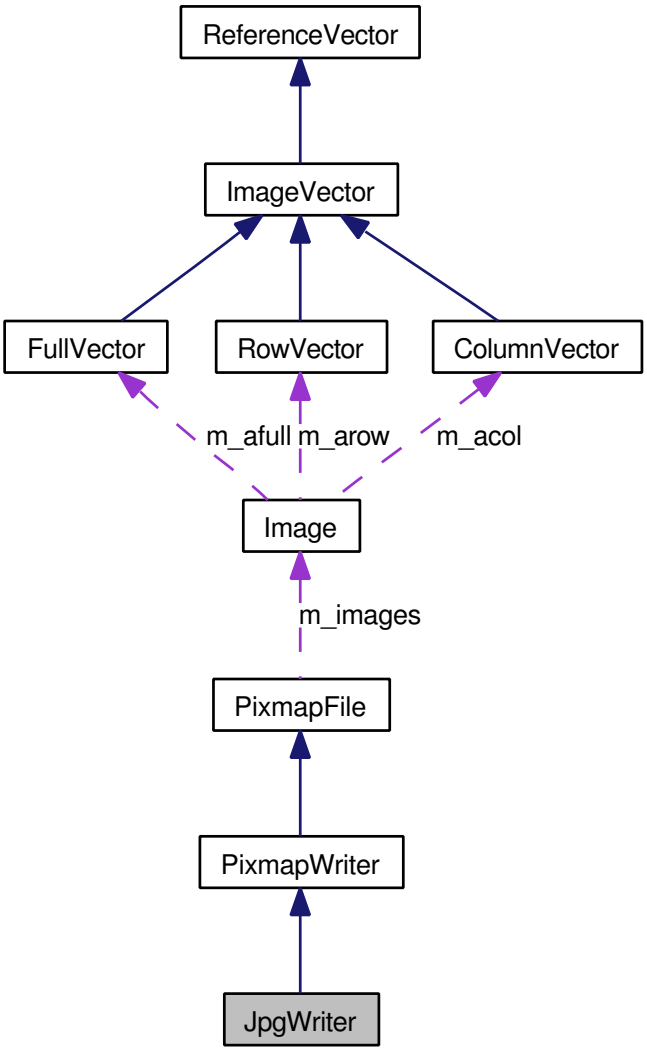
8.35 JpgWriter Class Reference

`#include <JpgWriter.hh>`

Inheritance diagram for JpgWriter:



Collaboration diagram for JpgWriter:



Public Member Functions

- JpgWriter (char const *name, Image *images[ ], int colors=3, int quality=90)

Protected Member Functions

- virtual int writefmt (void)

Protected Attributes

- int m_quality

8.35.1 Detailed Description

A JPG image file writer. Dimensions are known from the {*Image*} object. Only JPG version 6 is supported.

Definition at line 22 of file JpgWriter.hh.

8.35.2 Constructor & Destructor Documentation

8.35.2.1 JpgWriter::JpgWriter (char const * *name*, Image * *images*[ ], int *colors* = 3, int *quality* = 90) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference and set the offset if necessary.

Parameters:

name the file name

images the array of greyscale images for the values

colors the number of colors

quality the JPEG quality

Definition at line 37 of file JpgWriter.hh.

References m_quality.

8.35.3 Member Function Documentation

8.35.3.1 virtual int JpgWriter::writefmt (void) [protected, virtual]

Write a JPG image file. This is the method that does the actual work. It will be called by the {*write* ()} method.

Returns:

0 if successful, -1 on write error.

Implements PixmapWriter.

8.35.4 Member Data Documentation

8.35.4.1 int JpgWriter::m_quality [protected]

The JPEG quality factor

Definition at line 48 of file JpgWriter.hh.

Referenced by JpgWriter().

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

- JpgWriter.hh

8.36 logvals Struct Reference

`#include <ImageComparison.hh>`

Public Attributes

- int val
- bool init

8.36.1 Detailed Description

A data structure for a cache of (log (i) / log (2)) entries.

Definition at line 32 of file ImageComparison.hh.

8.36.2 Member Data Documentation

8.36.2.1 int logvals::val

Definition at line 33 of file ImageComparison.hh.

8.36.2.2 bool logvals::init

Definition at line 34 of file ImageComparison.hh.

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

- ImageComparison.hh

8.37 lq Struct Reference

`#include <ImageComparison.hh>`

Public Attributes

- double colors
- double details

8.37.1 Detailed Description

A data structure for an L^q norm.

Definition at line 24 of file ImageComparison.hh.

8.37.2 Member Data Documentation

8.37.2.1 double lq::colors

The weighted distance between the average colors

Definition at line 26 of file ImageComparison.hh.

8.37.2.2 double lq::details

The weighted distances between the top coefficients

Definition at line 28 of file ImageComparison.hh.

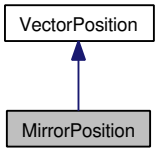
The documentation for this struct was generated from the following file:

- ImageComparison.hh

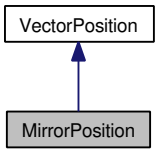
8.38 MirrorPosition Class Reference

`#include <MirrorPosition.hh>`

Inheritance diagram for MirrorPosition:



Collaboration diagram for MirrorPosition:



Public Member Functions

- MirrorPosition (bool laxis=false, bool raxis=false)
- MirrorPosition (int newsize)
- int pos (int position, int start, int end, int &sign)
- void setlaxis (bool laxis)
- void setraxis (bool raxis)
- bool laxis (void)
- bool raxis (void)

Private Member Functions

- int noaxis (int position, int start, int length, int &sign)
- int twoaxis (int position, int start, int length, int &sign)
- int oneleftaxis (int position, int start, int length, int &sign)
- int onerightaxis (int position, int start, int length, int &sign)

Private Attributes

- bool m_leftaxis
- bool m_rightaxis

8.38.1 Detailed Description

A Mirror index (abstract). Performs symmetric, periodic and whatever extensions to a vector or a part of it.

Definition at line 23 of file MirrorPosition.hh.

8.38.2 Constructor & Destructor Documentation

8.38.2.1 MirrorPosition::MirrorPosition (bool *laxis* = `false`, bool *raxis* = `false`)

Constructor. Allows to specify nonstandard symmetry details.

Parameters:

laxis an additional mirroring axis is used on the left side, like 3-2-1-0-|-0-1-2-3

raxis an additional mirroring axis is used on the right side, like 3-2-1-0-|-0-1-2-3

8.38.2.2 MirrorPosition::MirrorPosition (int *newsize*) `[inline]`

Constructor. Calls the superclass' constructor.

Definition at line 37 of file MirrorPosition.hh.

References m_leftaxis, and m_rightaxis.

8.38.3 Member Function Documentation

8.38.3.1 int MirrorPosition::pos (int *position*, int *start*, int *end*, int & *sign*) `[virtual]`

Calculate real position assuming a given vector start and length. If a position is greater than the assumed size or less than the assumed start position we mirror the position back into range.

Parameters:

position the requested position

start the assumed start

end the assumed end point

sign -1 if {*symmetry*} is set and we're in the first half of a period, else 1

Returns:

the new position

Implements VectorPosition.

8.38.3.2 void MirrorPosition::setlaxis (bool *laxis*) [inline]

Set left mirroring axis.

Parameters:

laxis an additional mirroring axis is used on the left side, like 3-2-1-0-|-0-1-2-3

Definition at line 60 of file MirrorPosition.hh.

References m_leftaxis.

8.38.3.3 void MirrorPosition::setraxis (bool *raxis*) [inline]

Set right mirroring axis.

Parameters:

raxis an additional mirroring axis is used on the reft side, like 0-1-2-3-|-3-2-1-0

Definition at line 66 of file MirrorPosition.hh.

References m_rightaxis.

8.38.3.4 bool MirrorPosition::laxis (void) [inline]

Return the left side axis settings.

Returns:

if an invisible axis exists: {*true}* else {*false}*

Definition at line 71 of file MirrorPosition.hh.

References m_leftaxis.

8.38.3.5 bool MirrorPosition::raxis (void) [inline]

Return the right side axis settings.

Returns:

if an invisible axis exists: {*true}* else {*false}*

Definition at line 75 of file MirrorPosition.hh.

References m_rightaxis.

8.38.3.6 int MirrorPosition::noaxis (int *position*, int *start*, int *length*, int & *sign*) [private]

Mirroring with the outer elements as axis on both sides.

Parameters:

position the requested position

start the assumed start

length the assumed length

sign -1 if {*m_symmetry}* is set and we're in the first half of a period, else 1

Returns:

the new position

8.38.3.7 int MirrorPosition::twoaxis (int *position*, int *start*, int *length*, int & *sign*) [private]

Mirroring with invisible axis on both sides.

Parameters:

position the requested position

start the assumed start

length the assumed length

sign -1 if {*m_symmetry}* is set and we're in the first half of a period, else 1

Returns:

the new position

8.38.3.8 int MirrorPosition::oneleftaxis (int *position*, int *start*, int *length*, int & *sign*) [private]

Mirroring with invisible axis on the left, but none on the right side.

Parameters:

position the requested position

start the assumed start

length the assumed length

sign -1 if {*m_symmetry}* is set and we're in the first half of a period, else 1

Returns:

the new position

8.38.3.9 int MirrorPosition::onerightaxis (int *position*, int *start*, int *length*, int & *sign*) [private]

Mirroring with invisible axis on the right, but none on the left side.

Parameters:

position the requested position

start the assumed start

length the assumed length

sign -1 if {*m_symmetry*} is set and we're in the first half of a period, else 1

Returns:

the new position

8.38.4 Member Data Documentation

8.38.4.1 bool MirrorPosition::m_leftaxis [private]

Invisible left axis for mirroring?

Definition at line 79 of file MirrorPosition.hh.

Referenced by laxis(), MirrorPosition(), and setlaxis().

8.38.4.2 bool MirrorPosition::m_rightaxis [private]

Invisible right axis for mirroring?

Definition at line 81 of file MirrorPosition.hh.

Referenced by MirrorPosition(), raxis(), and setraxis().

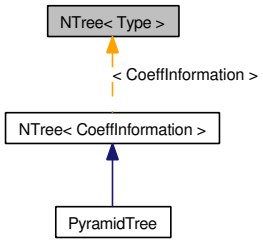
The documentation for this class was generated from the following file:

- MirrorPosition.hh

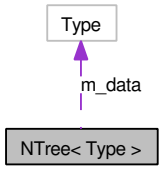
8.39 NTree< Type > Class Template Reference

`#include <NTree.hh>`

Inheritance diagram for NTree< Type >:



Collaboration diagram for NTree< Type >:



Public Member Functions

- NTree (int nChildren, NTree< Type > *parent=NULL, int position=-1)
- NTree (int nChildren, Type &data, int position=-1, NTree< Type > *parent=NULL)
- ~NTree (void)
- bool isRoot (void) const
- int position (void) const
- int aryness (void) const
- bool hasLeftSibling (void) const
- bool hasRightSibling (void) const
- bool hasChildAt (int pos) const
- bool hasChildren (void) const
- int card (void) const
- bool equals (const NTree &tree) const
- NTree< Type > * clone (void) const
- void destroyAt (int pos)
- void appendAt (int pos, NTree< Type > *tree)
- void appendAt (int pos, const Type &data)
- Type & data (void)
- NTree< Type > & childAt (int pos)
- NTree< Type > & leftSibling (void)
- NTree< Type > & rightSibling (void)

Protected Member Functions

- void init (int nChildren, int position, NTree< Type > *parent)
- void appendNGenerations (int levels)
- bool indexOK (int pos) const
- void copyLeaves (NTree< Type > &dst, const NTree< Type > &src) const

Protected Attributes

- NTree< Type > ** m_children
- NTree< Type > * m_parent
- Type m_data
- int m_aryness
- int m_position

8.39.1 Detailed Description

template<class Type> class NTree< Type >

A class for n-ary trees. Rather relaxed attitude towards position checking; crashes are avoided but errors not always flagged!

Definition at line 28 of file NTree.hh.

8.39.2 Constructor & Destructor Documentation

8.39.2.1 template<class Type> NTree< Type >::NTree (int *nChildren*, NTree< Type > * *parent* = NULL, int *position* = -1)

Constructor, creates one empty tree node. The data is uninitialized after creation.

Parameters:

nChildren the number of children to this node

position the position in the parent node if applicable

parent the parent node or NULL if root node

8.39.2.2 template<class Type> NTree< Type >::NTree (int *nChildren*, Type & *data*, int *position* = -1, NTree< Type > * *parent* = NULL)

Constructor, creates and initializes one tree node.

Parameters:

nChildren the number of children to this node

data the data to initialize with (the data will be assigned using the assignment operator)

position the position in the parent node if applicable

parent the parent node or NULL if root node

8.39.2.3 template<class Type> NTree< Type >::~NTree (void)

Destructor. Releases memory.

8.39.3 Member Function Documentation

8.39.3.1 template<class Type> bool NTree< Type >::isRoot (void) const [inline]

Is this node the root of a tree?

Returns:

true if root, false if not

Definition at line 50 of file NTree.hh.

Referenced by NTree< CoeffInformation >::hasLeftSibling(), and NTree< CoeffInformation >::hasRightSibling().

8.39.3.2 template<class Type> int NTree< Type >::position (void) const [inline]

Returns the position of this node in the parent node.

Returns:

the position in the parent node, -1 if this is a root node

Definition at line 53 of file NTree.hh.

8.39.3.3 template<class Type> int NTree< Type >::aryness (void) const [inline]

Returns the kind of tree this is (i.e. how many children each node has).

Returns:

the aryness

Definition at line 56 of file NTree.hh.

8.39.3.4 template<class Type> bool NTree< Type >::hasLeftSibling (void) const [inline]

Has this node any neighbour nodes to its left?

Returns:

true if there are

Definition at line 59 of file NTree.hh.

Referenced by NTree< CoeffInformation >::leftSibling().

8.39.3.5 template<class Type> bool NTree< Type >::hasRightSibling (void) const [inline]

Has this node any neighbour nodes to its right?

Returns:

true if there are

Definition at line 65 of file NTree.hh.

Referenced by NTree< CoeffInformation >::rightSibling().

8.39.3.6 template<class Type> bool NTree< Type >::hasChildAt (int *pos*) const [inline]

Has this node any children at a particular position?

Parameters:

pos the position

Returns:

true if it has any children at that position

Definition at line 72 of file NTree.hh.

Referenced by NTree< CoeffInformation >::hasChildren().

8.39.3.7 template<class Type> bool NTree< Type >::hasChildren (void) const [inline]

Has this node any children at all?

Returns:

true if it has children

Definition at line 77 of file NTree.hh.

8.39.3.8 template<class Type> int NTree< Type >::card (void) const

Returns the number of nodes.

Returns:

the number of nodes

8.39.3.9 template<class Type> bool NTree< Type >::equals (const NTree< Type > & *tree*) const

Compares two trees. The '==' equality operator will be used for the data fields

Parameters:

tree the other tree

Returns:

true if the trees have identical data / number of children

8.39.3.10 template<class Type> NTree< Type >* NTree< Type >::clone (void) const

Produce a copy of this tree. The data fields will be copied.

Returns:

the copy

8.39.3.11 template<class Type> void NTree< Type >::destroyAt (int *pos*) [inline]

Destroy the subtree at a particular position.

Parameters:

pos the position

Definition at line 94 of file NTree.hh.

8.39.3.12 template<class Type> void NTree< Type >::appendAt (int *pos*, NTree< Type > * *tree*)

Appends a tree at a particular position. The tree will not be duplicated.

Parameters:

pos the position

tree the tree

Exceptions:

invalid_argument if the tree has a different aryness from this

invalid_argument if the position is not empty

8.39.3.13 template<class Type> void NTree< Type >::appendAt (int *pos*, const Type & *data*)

Creates a new node and appends it at a particular position.

Parameters:

pos the position

data the data to append

Exceptions:

invalid_argument if the position is not empty

8.39.3.14 template<class Type> Type& NTree< Type >::data (void) [inline]

Return a reference to this node's data.

Returns:

a reference to the data

Definition at line 109 of file NTree.hh.

8.39.3.15 template<class Type> NTree< Type >& NTree< Type >::childAt (int *pos*) [inline]

Return a reference to the node/subtree at a particular position.

Parameters:

pos the position

Returns:

the subtree

Exceptions:

invalid_argument if the given position is empty

Definition at line 114 of file NTree.hh.

8.39.3.16 template<class Type> NTree< Type >& NTree< Type >::leftSibling (void) [inline]

Return a reference to the node/subtree left to this.

Returns:

the left sibling

Exceptions:

invalid_argument if there is no left sibling

Definition at line 124 of file NTree.hh.

8.39.3.17 template<class Type> NTree< Type >& NTree< Type >::rightSibling (void) [inline]

Return a reference to the node/subtree right to this.

Returns:

the right sibling

Exceptions:

invalid_argument if there is no right sibling

Definition at line 134 of file NTree.hh.

8.39.3.18 template<class Type> void NTree< Type >::init (int *nChildren*, int *position*, NTree< Type > * *parent*) [protected]

Initialize the node.

Parameters:

nChildren the number of children (aryness)

position the position in the parent node if applicable

parent the parent node or NULL if root

8.39.3.19 template<class Type> void NTree< Type >::appendNGenerations (int *levels*) [protected]

Create empty child nodes to this.

Parameters:

levels the number of generations to add.

8.39.3.20 template<class Type> bool NTree< Type >::indexOK (int *pos*) const [inline, protected]

Check whether an index is out of bounds.

Parameters:

pos the index

Returns:

false if the index is out of bounds

Definition at line 154 of file NTree.hh.

Referenced by NTree< CoeffInformation >::childAt(), and NTree< CoeffInformation >::hasChildAt().

8.39.3.21 template<class Type> void NTree< Type >::copyLeaves (NTree< Type > & *dst*, const NTree< Type > & *src*) const [protected]

Copy one tree's leaves to another tree. Data fields will be copied using the '=' assignment operator. Both trees need to have identical arynesses.

Parameters:

dst the destination tree

src the source tree

Exceptions:

invalid_argument if the dimensions do not match

8.39.4 Member Data Documentation

8.39.4.1 template<class Type> NTree< Type >** NTree< Type >::m_children [protected]

An array of 'aryness' elements holding the children's addresses

Definition at line 164 of file NTree.hh.

Referenced by NTree< CoeffInformation >::childAt(), NTree< CoeffInformation >::destroyAt(), and NTree< CoeffInformation >::hasChildAt().

8.39.4.2 template<class Type> NTree< Type >* NTree< Type >::m_parent [protected]

The parent node's address or NULL if root

Definition at line 167 of file NTree.hh.

Referenced by NTree< CoeffInformation >::hasLeftSibling(), NTree< CoeffInformation >::hasRightSibling(), NTree< CoeffInformation >::isRoot(), NTree< CoeffInformation >::leftSibling(), and NTree< CoeffInformation >::rightSibling().

8.39.4.3 template<class Type> Type NTree< Type >::m_data [protected]

The data field.

Definition at line 170 of file NTree.hh.

Referenced by NTree< CoeffInformation >::data().

8.39.4.4 template<class Type> int NTree< Type >::m_aryness [protected]

The number of children this node can have

Definition at line 173 of file NTree.hh.

Referenced by NTree< CoeffInformation >::aryness(), NTree< CoeffInformation >::hasChildren(), NTree< CoeffInformation >::hasRightSibling(), and NTree< CoeffInformation >::indexOK().

8.39.4.5 template<class Type> int NTree< Type >::m_position [protected]

This node's position in the parent node or -1 if root

Definition at line 175 of file NTree.hh.

Referenced by NTree< CoeffInformation >::hasLeftSibling(), NTree< CoeffInformation >::hasRightSibling(), NTree< CoeffInformation >::leftSibling(), NTree< CoeffInformation >::position(), and NTree< CoeffInformation >::rightSibling().

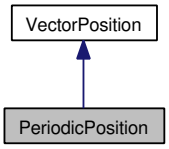
The documentation for this class was generated from the following file:

- NTree.hh

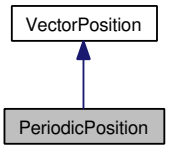
8.40 PeriodicPosition Class Reference

`#include <PeriodicPosition.hh>`

Inheritance diagram for PeriodicPosition:



Collaboration diagram for PeriodicPosition:



Public Member Functions

- PeriodicPosition (void)
- PeriodicPosition (int newsize)
- int pos (int pos, int start, int end, int &sign)

8.40.1 Detailed Description

A Periodic index (abstract). Performs symmetric, periodic and whatever extensions to a vector or a part of it.

Definition at line 23 of file PeriodicPosition.hh.

8.40.2 Constructor & Destructor Documentation

8.40.2.1 PeriodicPosition::PeriodicPosition (void) [inline]

Constructor. Calls the superclass' constructor.

Definition at line 28 of file PeriodicPosition.hh.

8.40.2.2 PeriodicPosition::PeriodicPosition (int *newsize*) [inline]

Constructor. Calls the superclass' constructor.

Definition at line 31 of file PeriodicPosition.hh.

8.40.3 Member Function Documentation

8.40.3.1 int PeriodicPosition::pos (int *pos*, int *start*, int *end*, int & *sign*) [virtual]

Calculate real position assuming a given vector start and length. If a position is greater than the assumed size or less than the assumed start position we mirror the position back into range.

Parameters:

pos the requested position

start the assumed start

end the assumed end point

sign always 1

Returns:

the new position

Implements VectorPosition.

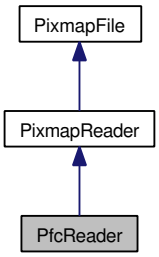
The documentation for this class was generated from the following file:

- PeriodicPosition.hh

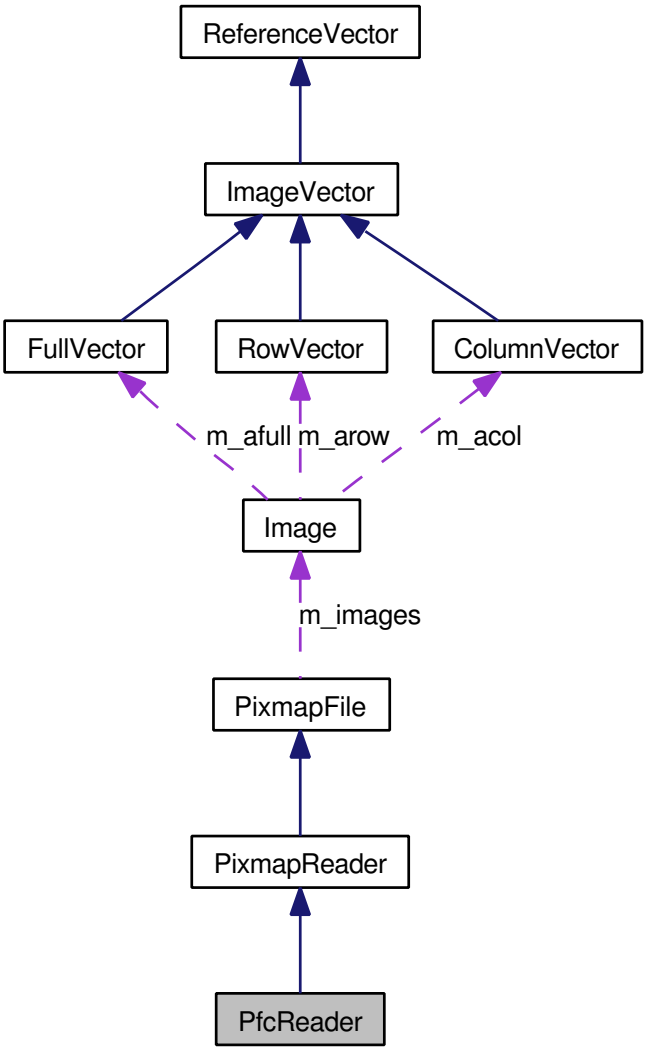
8.41 PfcReader Class Reference

`#include <PfcReader.hh>`

Inheritance diagram for PfcReader:



Collaboration diagram for PfcReader:



Public Member Functions

- PfcReader (char const *name, Image *images[ ], int colors=3)
- virtual ~PfcReader (void)

Protected Member Functions

- virtual int readfmt (void)

8.41.1 Detailed Description

A PFC file reader. Only PFCs version 6 are supported.

Definition at line 21 of file PfcReader.hh.

8.41.2 Constructor & Destructor Documentation

8.41.2.1 PfcReader::PfcReader (char const * *name*, Image * *images*[ ], int *colors* = 3) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference.

Parameters:

name the file name

images the array of greyscale images for the values

colors the number of colors

Definition at line 33 of file PfcReader.hh.

8.41.2.2 virtual PfcReader::~PfcReader (void) [inline, virtual]

Destructor - does nothing.

Definition at line 37 of file PfcReader.hh.

8.41.3 Member Function Documentation

8.41.3.1 virtual int PfcReader::readfmt (void) [protected, virtual]

Read the PFC file format. This does the actual work of reading and parsing the image file. It gets called by the {*read* ()} method.

Returns:

0 if successful, -1 on read error, -2 on file format error.

Implements PixmapReader.

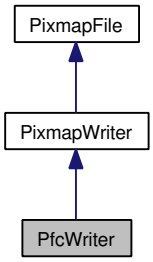
The documentation for this class was generated from the following file:

- PfcReader.hh

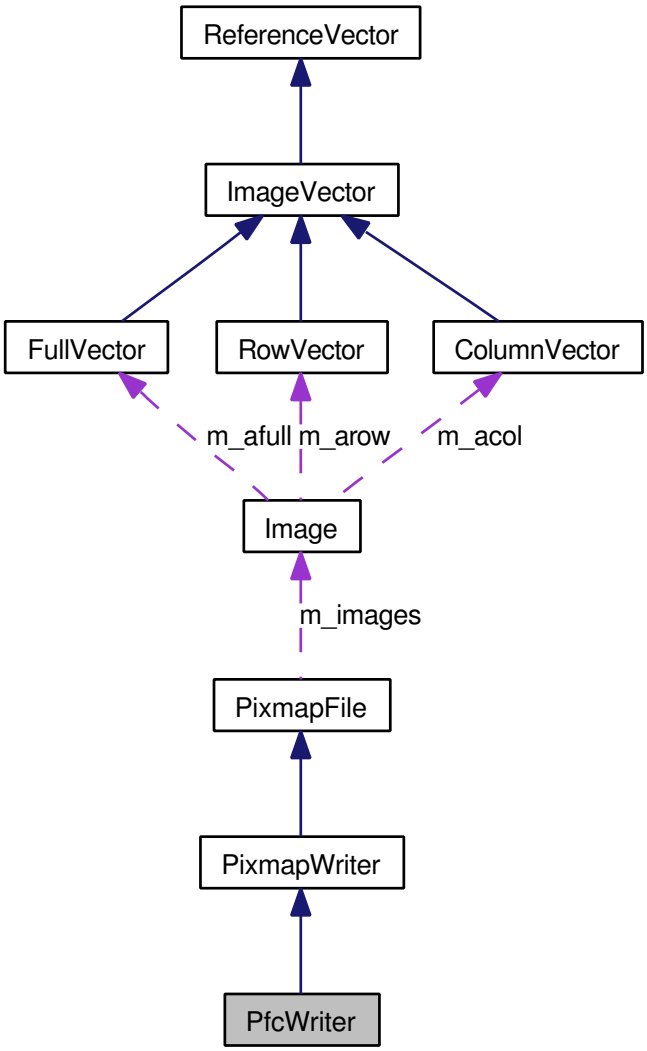
8.42 PfcWriter Class Reference

`#include <PfcWriter.hh>`

Inheritance diagram for PfcWriter:



Collaboration diagram for PfcWriter:



Public Member Functions

- PfcWriter (char const *name, Image *images[ ], int colors=3, clrmodel cmodel=cm_rgb)
- virtual ~PfcWriter (void)

Protected Member Functions

- virtual int writefmt (void)

8.42.1 Detailed Description

A PFC image file writer. Dimensions are known from the {*Image*} object. Only PFC version 6 is supported.

Definition at line 21 of file PfcWriter.hh.

8.42.2 Constructor & Destructor Documentation

8.42.2.1 PfcWriter::PfcWriter (char const * *name*, Image * *images*[ ], int *colors* = 3, clrmodel *cmodel* = cm_rgb) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference and set the offset if necessary.

Parameters:

name the file name

images the array of greyscale images for the values

colors the number of colors

cmodel the color model

Definition at line 36 of file PfcWriter.hh.

References PixmapFile::colormodel().

8.42.2.2 virtual PfcWriter::~PfcWriter (void) [inline, virtual]

Destructor - does nothing.

Definition at line 41 of file PfcWriter.hh.

8.42.3 Member Function Documentation

8.42.3.1 virtual int PfcWriter::writefmt (void) [protected, virtual]

Write a PFC image file. This is the method that does the actual work. It will be called by the {*write* ()} method.

Returns:

0 if successful, -1 on write error.

Implements PixmapWriter.

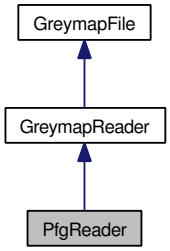
The documentation for this class was generated from the following file:

- PfcWriter.hh

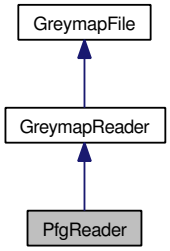
8.43 PfgReader Class Reference

`#include <PfgReader.hh>`

Inheritance diagram for PfgReader:



Collaboration diagram for PfgReader:



Public Member Functions

- PfgReader (char const *name, ImageArray< coeff > &data)
- virtual ~PfgReader (void)

Protected Member Functions

- virtual int readfmt (void)

8.43.1 Detailed Description

A PFG file reader. Only grey-scale PFGs are supported.

Definition at line 21 of file PfgReader.hh.

8.43.2 Constructor & Destructor Documentation

8.43.2.1 PfgReader::PfgReader (char const * *name*, ImageArray< coeff > & *data*) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference.

Parameters:

name the file name

data the reference to the {*ImageArray*} object

Definition at line 31 of file PfgReader.hh.

8.43.2.2 virtual PfgReader::~PfgReader (void) [inline, virtual]

Destructor - does nothing.

Definition at line 35 of file PfgReader.hh.

8.43.3 Member Function Documentation

8.43.3.1 virtual int PfgReader::readfmt (void) [protected, virtual]

Read the PFG file format. This does the actual work of reading and parsing the image file. It gets called by the {*read* ()} method.

Returns:

0 if successful, -1 on read error, -2 on file format error.

Implements GreymapReader.

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

- PfgReader.hh

8.44 PfgWriter Class Reference

`#include <PfgWriter.hh>`

Inheritance diagram for PfgWriter:

Collaboration diagram for PfgWriter:

Public Member Functions

- PfgWriter (char const *name, ImageArray< coeff > &data)
- virtual ~PfgWriter (void)

Protected Member Functions

- virtual int writefmt (void)

8.44.1 Detailed Description

A PFG image file writer. Dimensions are known from the {*ImageArray*} object. Only grey-scale PFGs are supported.

Definition at line 22 of file PfgWriter.hh.

8.44.2 Constructor & Destructor Documentation

8.44.2.1 PfgWriter::PfgWriter (char const * *name*, ImageArray< coeff > & *data*) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference and set the offset if necessary.

Parameters:

name the file name

data the reference to the {*ImageArray*} object

Definition at line 33 of file PfgWriter.hh.

8.44.2.2 virtual PfgWriter::~PfgWriter (void) [inline, virtual]

Destructor - does nothing.

Definition at line 37 of file PfgWriter.hh.

8.44.3 Member Function Documentation

8.44.3.1 virtual int PfgWriter::writefmt (void) [protected, virtual]

Write a PFG image file. This is the method that does the actual work. It will be called by the {*write* ()} method.

Returns:

0 if successful, -1 on write error.

Implements GreymapWriter.

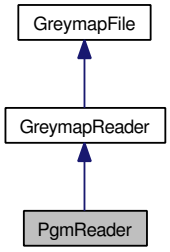
The documentation for this class was generated from the following file:

- PfgWriter.hh

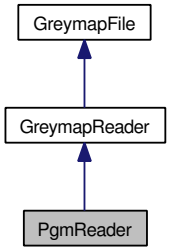
8.45 PgmReader Class Reference

`#include <PgmReader.hh>`

Inheritance diagram for PgmReader:



Collaboration diagram for PgmReader:



Public Member Functions

- PgmReader (char const *name, ImageArray< coeff > &data)
- virtual ~PgmReader (void)

Protected Member Functions

- virtual int readfmt (void)

8.45.1 Detailed Description

A PGM file reader. Only PGMs version 5 are supported.

Definition at line 21 of file PgmReader.hh.

8.45.2 Constructor & Destructor Documentation

8.45.2.1 PgmReader::PgmReader (char const * *name*, ImageArray< coeff > & *data*) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference.

Parameters:

name the file name

data the reference to the {*ImageArray*} object

Definition at line 31 of file PgmReader.hh.

8.45.2.2 virtual PgmReader::~PgmReader (void) [inline, virtual]

Destructor - does nothing.

Definition at line 35 of file PgmReader.hh.

8.45.3 Member Function Documentation

8.45.3.1 virtual int PgmReader::readfmt (void) [protected, virtual]

Read the PGM file format. This does the actual work of reading and parsing the image file. It gets called by the {*read* ()} method.

Returns:

0 if successful, -1 on read error, -2 on file format error.

Implements GreymapReader.

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

- PgmReader.hh

8.46 PgmWriter Class Reference

`#include <PgmWriter.hh>`

Inheritance diagram for PgmWriter:

Collaboration diagram for PgmWriter:

Public Member Functions

- PgmWriter (char const *name, ImageArray< coeff > &data)
- virtual ~PgmWriter (void)

Protected Member Functions

- virtual int writefmt (void)

8.46.1 Detailed Description

A PGM image file writer. Dimensions are known from the {*ImageArray*} object. Only PGM version 5 is supported.

Definition at line 22 of file PgmWriter.hh.

8.46.2 Constructor & Destructor Documentation

8.46.2.1 PgmWriter::PgmWriter (char const * *name*, ImageArray< coeff > & *data*) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference and set the offset if necessary.

Parameters:

name the file name

data the reference to the {*ImageArray*} object

Definition at line 33 of file PgmWriter.hh.

8.46.2.2 virtual PgmWriter::~PgmWriter (void) [inline, virtual]

Destructor - does nothing.

Definition at line 37 of file PgmWriter.hh.

8.46.3 Member Function Documentation

8.46.3.1 virtual int PgmWriter::writefmt (void) [protected, virtual]

Write a PGM image file. This is the method that does the actual work. It will be called by the {*write* ()} method.

Returns:

0 if successful, -1 on write error.

Implements GreymapWriter.

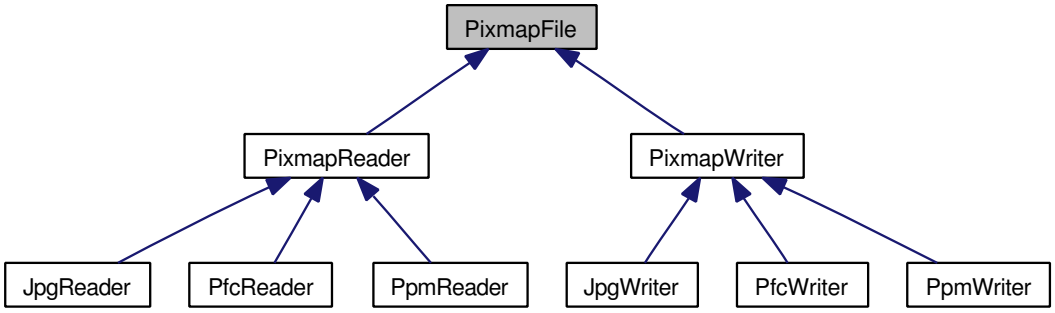
The documentation for this class was generated from the following file:

- PgmWriter.hh

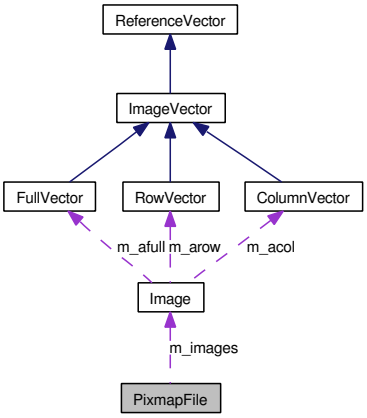
8.47 PixmapFile Class Reference

`#include <PixmapFile.hh>`

Inheritance diagram for PixmapFile:



Collaboration diagram for PixmapFile:



Public Member Functions

- PixmapFile (char const *name, Image *images[ ], int channels)
- virtual ~PixmapFile (void)
- clrmodel colormodel (void)
- void colormodel (clrmodel cm)
- int channels (void) const

Protected Member Functions

- void init (Image *images[ ])

Protected Attributes

- Image ** m_images
- int m_channels
- char const * m_fname
- clrmodel m_cmodel

8.47.1 Detailed Description

An abstract image file. A framework to create readers and writers on any color images.

Definition at line 24 of file PixmapFile.hh.

8.47.2 Constructor & Destructor Documentation

8.47.2.1 PixmapFile::PixmapFile (char const * *name*, Image * *images*[ ], int *channels*)

Constructor. Initializes internal fields and gets an {*ImageArray*} object that may already contain an image or will get one later.

Parameters:

name the file name

images the array of greyscale images for the values

channels the number of colors

8.47.2.2 virtual PixmapFile::~PixmapFile (void) [virtual]

Destructor. Releases some memory.

8.47.3 Member Function Documentation

8.47.3.1 clrmodel PixmapFile::colormodel (void) [inline]

Returns the current color model.

Returns:

the current color model

Definition at line 44 of file PixmapFile.hh.

Referenced by PfcWriter::PfcWriter().

8.47.3.2 void PixmapFile::colormodel (clrmodel *cm*) [inline]

Sets a new color model.

Parameters:

cm the new color model

Definition at line 47 of file PixmapFile.hh.

8.47.3.3 int PixmapFile::channels (void) const [inline]

Return the number of channels.

Returns:

the number of channels

Definition at line 51 of file PixmapFile.hh.

8.47.3.4 void PixmapFile::init (Image * *images*[ ]) [protected]

internal initialization

8.47.4 Member Data Documentation

8.47.4.1 Image** PixmapFile::m_images [protected]

The color channels.

Definition at line 56 of file PixmapFile.hh.

8.47.4.2 int PixmapFile::m_channels [protected]

The number of channels (usually: colors).

Definition at line 58 of file PixmapFile.hh.

8.47.4.3 char const* PixmapFile::m_fname [protected]

The file name. The name of the file associated with this object.

Definition at line 60 of file PixmapFile.hh.

8.47.4.4 clrmodel PixmapFile::m_cmodel [protected]

Definition at line 62 of file PixmapFile.hh.

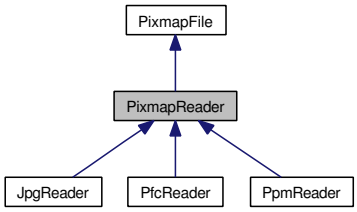
The documentation for this class was generated from the following file:

- PixmapFile.hh

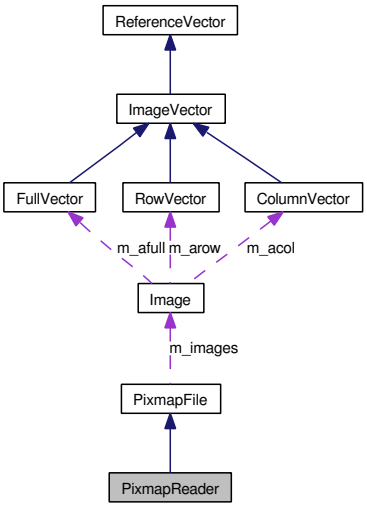
8.48 PixmapReader Class Reference

`#include <PixmapReader.hh>`

Inheritance diagram for PixmapReader:



Collaboration diagram for PixmapReader:



Public Member Functions

- PixmapReader (char const *name, Image *images[ ], int channels)
- virtual ~PixmapReader (void)
- virtual void read (void)

Protected Member Functions

- virtual int readfmt (void)=0

8.48.1 Detailed Description

An abstract color image reader. The image is being read putting its color channels into separate greyscale image objects.

Definition at line 24 of file PixmapReader.hh.

8.48.2 Constructor & Destructor Documentation

8.48.2.1 PixmapReader::PixmapReader (char const * *name*, Image * *images*[ ], int *channels*) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference.

Parameters:

name the file name

images the array of greyscale images for the values

channels the number of colors

Definition at line 37 of file PixmapReader.hh.

8.48.2.2 virtual PixmapReader::~PixmapReader (void) [inline, virtual]

Destructor - does nothing.

Definition at line 41 of file PixmapReader.hh.

8.48.3 Member Function Documentation

8.48.3.1 virtual void PixmapReader::read (void) [virtual]

Read the image. All steps independent of the file format will be performed, like testing for file readability etc.

Exceptions:

invalid_argument invalid file format

ios_base::failure a read error has occured [not supported by all libraries, so eventually {*invalid_argument*} instead]

8.48.3.2 virtual int PixmapReader::readfmt (void) [protected, pure virtual]

Read different file formats (abstract). This is the method to be implemented for every image file format. It will be called by the {*read* ()} method.

Returns:

0 if successful, -1 on read error, -2 on file format error.

Implemented in JpgReader, PfcReader, and PpmReader.

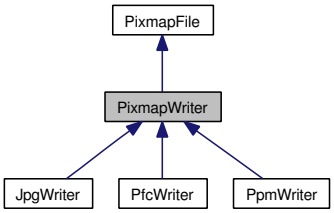
The documentation for this class was generated from the following file:

- PixmapReader.hh

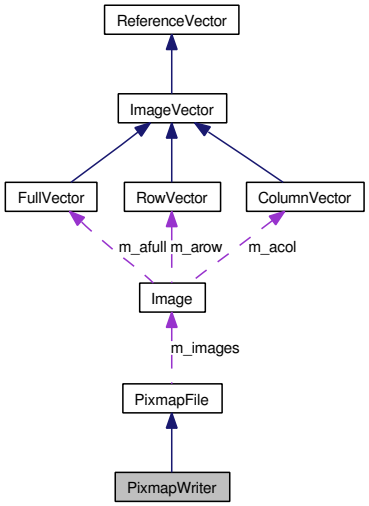
8.49 PixmapWriter Class Reference

`#include <PixmapWriter.hh>`

Inheritance diagram for PixmapWriter:



Collaboration diagram for PixmapWriter:



Public Member Functions

- PixmapWriter (char const *name, Image *images[ ], int channels)
- virtual ~PixmapWriter (void)
- void write (void)

Protected Member Functions

- virtual int writefmt (void)=0

8.49.1 Detailed Description

An abstract image writer. The image is being written putting its content from an array of greyscale image objects into the file.

Definition at line 23 of file PixmapWriter.hh.

8.49.2 Constructor & Destructor Documentation

8.49.2.1 PixmapWriter::PixmapWriter (char const * *name*, Image * *images*[ ], int *channels*) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference.

Parameters:

name the file name

images the array of greyscale images for the values

channels the number of colors

Definition at line 36 of file PixmapWriter.hh.

8.49.2.2 virtual PixmapWriter::~PixmapWriter (void) [inline, virtual]

Destructor - does nothing.

Definition at line 40 of file PixmapWriter.hh.

8.49.3 Member Function Documentation

8.49.3.1 void PixmapWriter::write (void)

Write the image. All steps independent of the file format will be performed, like testing for file writeability etc.

Exceptions:

ios_base::failure a read error has occured [not supported by all libraries, so eventually {*invalid_argument*} instead]

8.49.3.2 virtual int PixmapWriter::writefmt (void) [protected, pure virtual]

Write different file formats (abstract). This is the method to be implemented for every image file format. It will be called by the {*write* ()} method.

Returns:

0 if successful, -1 on write error.

Implemented in JpgWriter, PfcWriter, and PpmWriter.

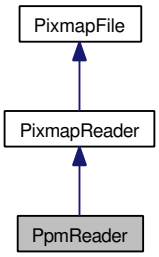
The documentation for this class was generated from the following file:

- PixmapWriter.hh

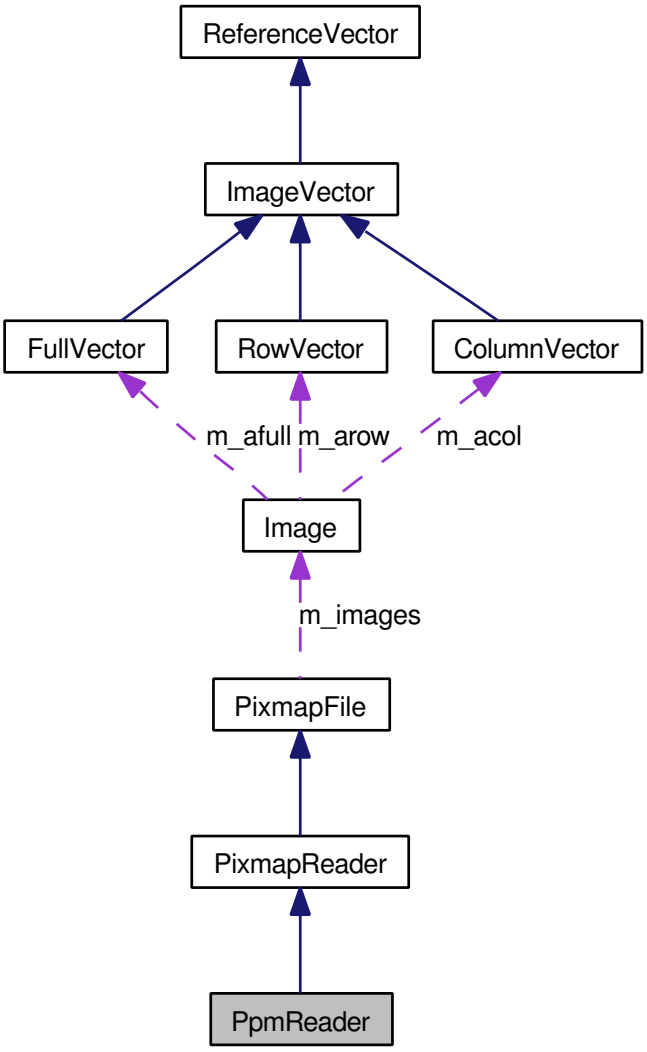
8.50 PpmReader Class Reference

`#include <PpmReader.hh>`

Inheritance diagram for PpmReader:



Collaboration diagram for PpmReader:



Public Member Functions

- PpmReader (char const *name, Image *images[ ])
- virtual ~PpmReader (void)

Protected Member Functions

- virtual int readfmt (void)

8.50.1 Detailed Description

A PPM file reader. Only PPMs version 6 are supported.

Definition at line 21 of file PpmReader.hh.

8.50.2 Constructor & Destructor Documentation

8.50.2.1 PpmReader::PpmReader (char const * *name*, Image * *images*[ ]) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference.

Parameters:

name the file name

images the array of greyscale images for the values

Definition at line 31 of file PpmReader.hh.

8.50.2.2 virtual PpmReader::~PpmReader (void) [inline, virtual]

Destructor - does nothing.

Definition at line 35 of file PpmReader.hh.

8.50.3 Member Function Documentation

8.50.3.1 virtual int PpmReader::readfmt (void) [protected, virtual]

Read the PPM file format. This does the actual work of reading and parsing the image file. It gets called by the {*read* ()} method.

Returns:

0 if successful, -1 on read error, -2 on file format error.

Implements PixmapReader.

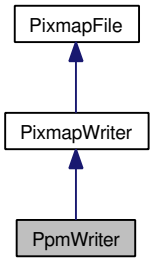
The documentation for this class was generated from the following file:

- PpmReader.hh

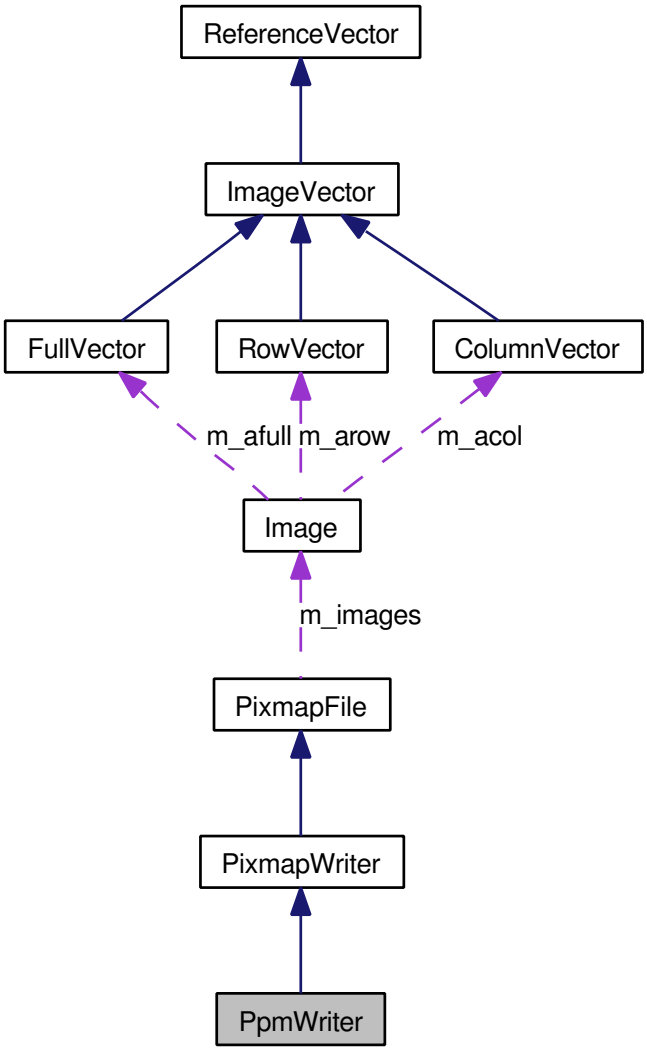
8.51 PpmWriter Class Reference

`#include <PpmWriter.hh>`

Inheritance diagram for PpmWriter:



Collaboration diagram for PpmWriter:



Public Member Functions

- PpmWriter (char const *name, Image *images[ ])
- virtual ~PpmWriter (void)

Protected Member Functions

- virtual int writefmt (void)

8.51.1 Detailed Description

A PPM image file writer. Dimensions are known from the {*Image*} object. Only PPM version 6 is supported.

Definition at line 22 of file PpmWriter.hh.

8.51.2 Constructor & Destructor Documentation

8.51.2.1 PpmWriter::PpmWriter (char const * *name*, Image * *images*[ ]) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference and set the offset if necessary.

Parameters:

name the file name

images the array of greyscale images for the values

Definition at line 33 of file PpmWriter.hh.

8.51.2.2 virtual PpmWriter::~PpmWriter (void) [inline, virtual]

Destructor - does nothing.

Definition at line 37 of file PpmWriter.hh.

8.51.3 Member Function Documentation

8.51.3.1 virtual int PpmWriter::writefmt (void) [protected, virtual]

Write a PPM image file. This is the method that does the actual work. It will be called by the {*write* ()} method.

Returns:

0 if successful, -1 on write error.

Implements PixmapWriter.

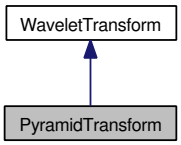
The documentation for this class was generated from the following file:

- PpmWriter.hh

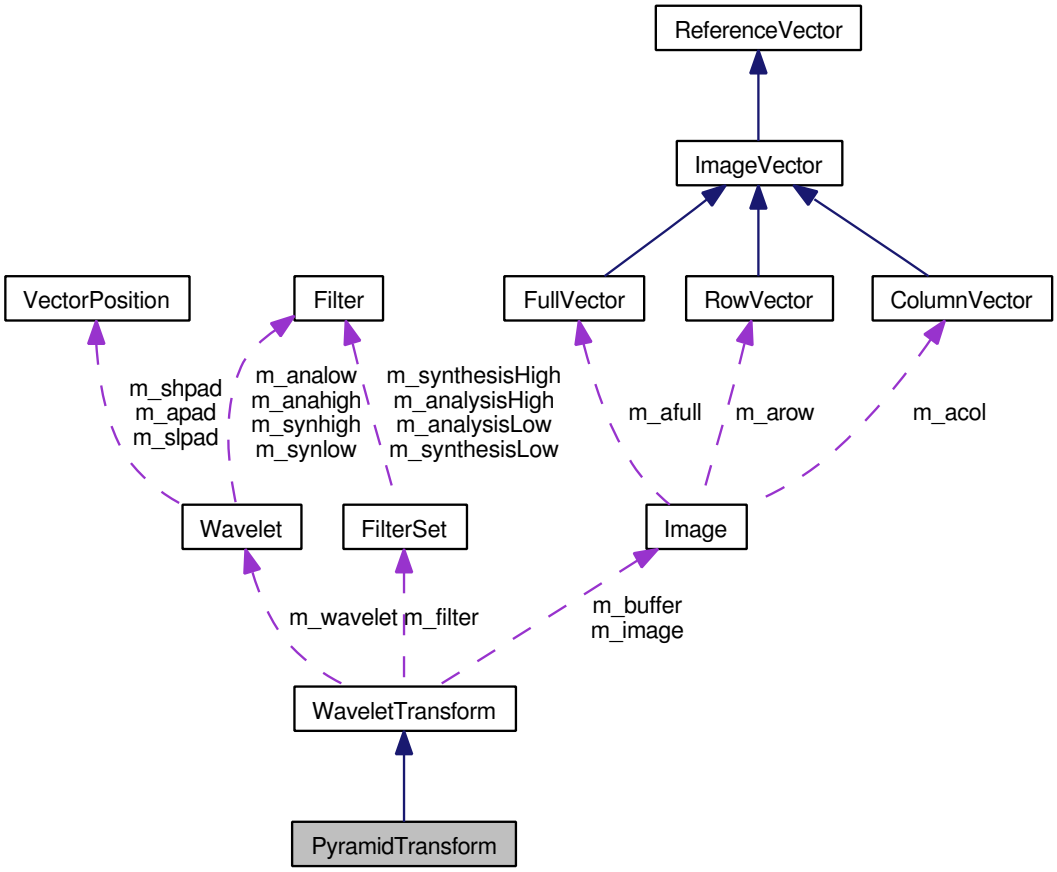
8.52 PyramidTransform Class Reference

`#include <PyramidTransform.hh>`

Inheritance diagram for PyramidTransform:



Collaboration diagram for PyramidTransform:



Public Member Functions

- PyramidTransform (Image &img, FilterSet &fil)

Protected Member Functions

- virtual void doanalysis (int steps)
- virtual void dosynthesis (int steps, int prevSteps=0)

8.52.1 Detailed Description

A Pyramid Transform. The transform is two-dimensional, it thus works on images.

Definition at line 23 of file PyramidTransform.hh.

8.52.2 Constructor & Destructor Documentation

8.52.2.1 PyramidTransform::PyramidTransform (Image & *img*, FilterSet & *fil*) [inline]

Constructor. Calls the superclass' constructor.

Parameters:

img the image

fil the filter (wavelet)

Definition at line 32 of file PyramidTransform.hh.

8.52.3 Member Function Documentation

8.52.3.1 virtual void PyramidTransform::doanalysis (int *steps*) [protected, virtual]

Perform a Pyramid transform on the image.

Parameters:

steps the number of transform steps

Implements WaveletTransform.

8.52.3.2 virtual void PyramidTransform::dosynthesis (int *steps*, int *prevSteps* = 0) [protected, virtual]

Perform an inverse Pyramid transform on the image.

Parameters:

steps the number of inverse transform steps

prevSteps if greater than 0, the number of previous decomposition steps to assume. This is necessary to reconstruct images that are not square and have side lengths which are not powers of two.

Implements WaveletTransform.

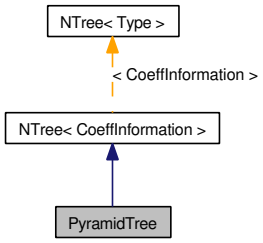
The documentation for this class was generated from the following file:

- PyramidTransform.hh

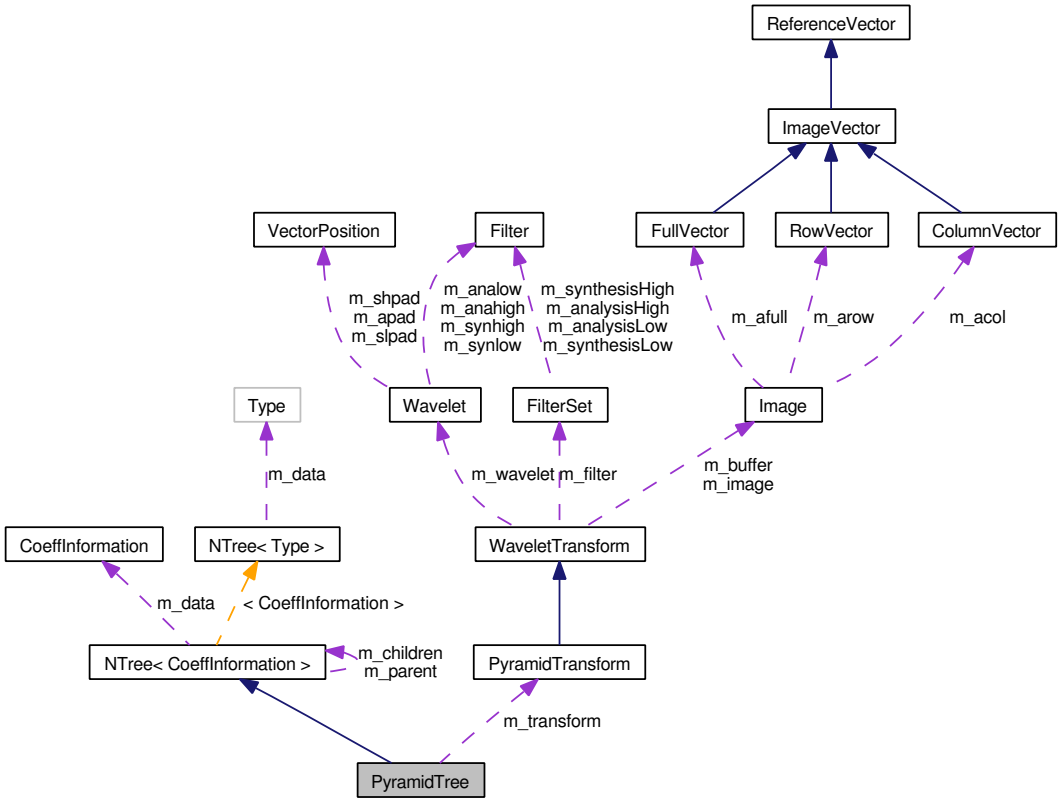
8.53 PyramidTree Class Reference

`#include <PyramidTree.hh>`

Inheritance diagram for PyramidTree:



Collaboration diagram for PyramidTree:



Public Member Functions

- PyramidTree (PyramidTransform &t, int y, int x, int position=-1, PyramidTree *parent=NULL, int generations=1)

- ~PyramidTree (void)
- void addGenerations (int depth)
- void moveTo (int y, int x, int position=-1, PyramidTree *parent=NULL)
- int depth (void)

Protected Member Functions

- void findChildPosition (int y, int x, int &resY, int &resX)
- void shiftBy (int shiftY, int shiftX)

Private Attributes

- PyramidTransform * m_transform

8.53.1 Detailed Description

A class for pyramid trees. Rather relaxed attitude towards position checking; crashes are avoided but errors not always flagged! Each node can have exactly 4 children. Child #0 is the upper left, the children are stored linewise from there.

Definition at line 26 of file PyramidTree.hh.

8.53.2 Constructor & Destructor Documentation

8.53.2.1 PyramidTree::PyramidTree (PyramidTransform & *t*, int *y*, int *x*, int *position* = -1, PyramidTree * *parent* = NULL, int *generations* = 1)

Constructor, creates a tree from a transformed image at a given position and over a depth of a given number of generations

Parameters:

t the transform

y the row in the image

x the col in the image

position the position in the parent node if applicable

parent a pointer to the parent node or NULL if root

generations the number of generations (including this node)

Exceptions:

invalid_argument if the transform levels are insufficient or the start position in the image's LL

8.53.2.2 PyramidTree::~PyramidTree (void)

Destructor, does nothing (destruction done by parent class)

8.53.3 Member Function Documentation

8.53.3.1 void PyramidTree::addGenerations (int *depth*)

Create/replace a number of generations to this node according to position and subband.

Exceptions:

invalid_argument if the subband depth is insufficient

Parameters:

depth the depth to recurse into (not counting this node)

8.53.3.2 void PyramidTree::moveTo (int *y*, int *x*, int *position* = `-1`, PyramidTree * *parent* = `NULL`)

"Move" the tree to a different position in the image by replacing the positions stored in the leaves.

Parameters:

y the new row

x the new column

position the new position in the parent node (or -1 if no change)

parent the new parent node (or NULL if no change)

8.53.3.3 int PyramidTree::depth (void)

Return the current depth including this node

Returns:

the depth

8.53.3.4 void PyramidTree::findChildPosition (int *y*, int *x*, int & *resY*, int & *resX*) `[protected]`

Find the upper left of the position for a new child

Parameters:

y the row in the image

x the col in the image

resY the returned row

resX the returned col

Exceptions:

invalid_argument if (y,x) is in subband 1 or in LL

8.53.3.5 void PyramidTree::shiftBy (int *shiftY*, int *shiftX*) [protected]

Shift the stored positions in the node and all children by a given amount.

Parameters:

shiftY the number of rows by which to shift

shiftX the number of columns by which to shift

8.53.4 Member Data Documentation

8.53.4.1 PyramidTransform* PyramidTree::m_transform [private]

The transform (contains a reference to the image)

Definition at line 76 of file PyramidTree.hh.

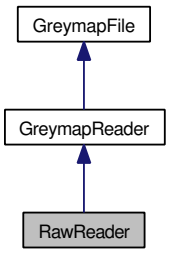
The documentation for this class was generated from the following file:

- PyramidTree.hh

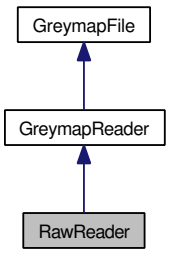
8.54 RawReader Class Reference

`#include <RawReader.hh>`

Inheritance diagram for RawReader:



Collaboration diagram for RawReader:



Public Member Functions

- RawReader (char const *name, ImageArray< coeff > &data, int y=512, int x=512, int offs=0)
- virtual ~RawReader (void)
- void header (int offs)
- void cols (int cols)
- void rows (int rows)

Protected Member Functions

- virtual int readfmt (void)

Private Attributes

- int m_offset
- int m_xsize
- int m_ysize

8.54.1 Detailed Description

A RAW file reader. Dimensions and a read offset can be specified through additional arguments to the constructor or through some methods.

Definition at line 21 of file RawReader.hh.

8.54.2 Constructor & Destructor Documentation

8.54.2.1 RawReader::RawReader (char const * *name*, ImageArray< coeff > & *data*, int *y* = 512, int *x* = 512, int *offs* = 0) [inline]

Constructor. Calls the mother class' constructor to initialize the {*ImageArray*} reference and eventually sets the image dimensions plus offset.

Parameters:

name the file name

data the reference to the {*ImageArray*} object

y the number of rows

x the number of cols

offs the offset

Definition at line 39 of file RawReader.hh.

References cols(), header(), and rows().

8.54.2.2 virtual RawReader::~RawReader (void) [inline, virtual]

Destructor - does nothing.

Definition at line 44 of file RawReader.hh.

8.54.3 Member Function Documentation

8.54.3.1 void RawReader::header (int *offs*) [inline]

Set the offset.

Parameters:

offs the new offset

Definition at line 49 of file RawReader.hh.

References m_offset.

Referenced by RawReader().

8.54.3.2 void RawReader::cols (int *cols*) [inline]

Set the horizontal size.

Parameters:

cols the new number of cols

Definition at line 53 of file RawReader.hh.

References m_xsize.

Referenced by RawReader().

8.54.3.3 void RawReader::rows (int *rows*) [inline]

Set the vertical size.

Parameters:

rows the new number of rows

Definition at line 57 of file RawReader.hh.

References m_ysize.

Referenced by RawReader().

8.54.3.4 virtual int RawReader::readfmt (void) [protected, virtual]

Read the RAW file format. This does the actual work of reading and parsing the image file. It gets called by the {*read* ()} method.

Returns:

0 if successful, -1 on read error, -2 on file format error.

Implements GreymapReader.

8.54.4 Member Data Documentation

8.54.4.1 int RawReader::m_offset [private]

The read offset. Some files have some leading bytes containing no usable information. If this applies this has a non-zero value.

Definition at line 62 of file RawReader.hh.

Referenced by header().

8.54.4.2 int RawReader::m_xsize [private]

The horizontal size.

Definition at line 64 of file RawReader.hh.

Referenced by cols().

8.54.4.3 int RawReader::m_ysize [private]

The vertical size.

Definition at line 66 of file RawReader.hh.

Referenced by rows().

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

- RawReader.hh

8.55 RawWriter Class Reference

`#include <RawWriter.hh>`

Inheritance diagram for RawWriter:

Collaboration diagram for RawWriter:

Public Member Functions

- RawWriter (char const *name, ImageArray< coeff > &data, int offs=0)
- virtual ~RawWriter (void)
- void header (int offs)

Protected Member Functions

- virtual int writefmt (void)

Private Attributes

- int m_offset

8.55.1 Detailed Description

A RAW image file writer. Dimensions are known from the {*ImageArray*} object.

Definition at line 22 of file RawWriter.hh.

8.55.2 Constructor & Destructor Documentation

8.55.2.1 RawWriter::RawWriter (char const * *name*, ImageArray< coeff > & *data*, int *offs* = 0) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference and set the offset if necessary.

Parameters:

name the file name

data the reference to the {*ImageArray*} object

offs the offset

Definition at line 35 of file RawWriter.hh.

References header().

8.55.2.2 virtual RawWriter::~RawWriter (void) [inline, virtual]

Destructor - does nothing.

Definition at line 39 of file RawWriter.hh.

8.55.3 Member Function Documentation

8.55.3.1 void RawWriter::header (int *offs*) [inline]

Set the offset.

Parameters:

offs the new offset

Definition at line 44 of file RawWriter.hh.

References m_offset.

Referenced by RawWriter().

8.55.3.2 virtual int RawWriter::writefmt (void) [protected, virtual]

Write a RAW image file. This is the method that does the actual work. It will be called by the {*write* ()} method.

Returns:

0 if successful, -1 on write error.

Implements GreymapWriter.

8.55.4 Member Data Documentation

8.55.4.1 int RawWriter::m_offset [private]

The read offset. Some files have some leading bytes containing no usable information. If this applies this has a non-zero value.

Definition at line 49 of file RawWriter.hh.

Referenced by header().

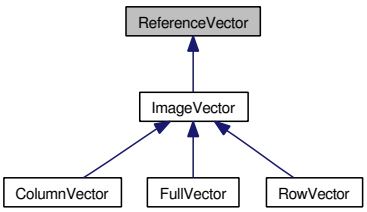
The documentation for this class was generated from the following file:

- RawWriter.hh

8.56 ReferenceVector Class Reference

`#include <ReferenceVector.hh>`

Inheritance diagram for ReferenceVector:



Public Member Functions

- ReferenceVector (void)
- virtual ~ReferenceVector (void)
- int root (void)
- virtual bool sanity (void)=0
- virtual void update (void)=0
- virtual void go (int root)=0
- virtual coeff at (int pos)=0
- virtual void to (int pos, coeff val)=0
- virtual int size (void)=0
- void copy (ReferenceVector &other)

Protected Attributes

- int m_vroot

8.56.1 Detailed Description

Pseudo-vector (abstract). Provide a two-dimensional array's rows and columns as vectors with array addressing.

Definition at line 23 of file ReferenceVector.hh.

8.56.2 Constructor & Destructor Documentation

8.56.2.1 ReferenceVector::ReferenceVector (void) `[inline]`

Constructor. Simple initialization.

Definition at line 29 of file ReferenceVector.hh.

8.56.2.2 virtual ReferenceVector::~ReferenceVector (void) [inline, virtual]

Destructor. Does nothing.

Definition at line 33 of file ReferenceVector.hh.

8.56.3 Member Function Documentation

8.56.3.1 int ReferenceVector::root (void) [inline]

Returns the root.

Definition at line 36 of file ReferenceVector.hh.

8.56.3.2 virtual bool ReferenceVector::sanity (void) [pure virtual]

Checks integrity. If the array's dimensions have been changed (resize operation) {*false*} will be returned.

Implemented in ImageVector.

8.56.3.3 virtual void ReferenceVector::update (void) [pure virtual]

Updates the vector's settings. This is necessary each time after the array has been resized.

Implemented in ColumnVector, FullVector, ImageVector, and RowVector.

8.56.3.4 virtual void ReferenceVector::go (int *root*) [pure virtual]

Set new root. Depending of what concrete instance is the current row or col will be set.

Exceptions:

invalid_argument a negative value was given for the new root

Parameters:

root the new root

Implemented in ColumnVector, FullVector, ImageVector, and RowVector.

8.56.3.5 virtual coeff ReferenceVector::at (int *pos*) [pure virtual]

Return a value (abstract). The value is taken from the vector's position {*pos*}.

Parameters:

pos the position

Returns:

the value at that position

Implemented in ColumnVector, FullVector, ImageVector, and RowVector.

Referenced by copy().

8.56.3.6 virtual void ReferenceVector::to (int *pos*, coeff *val*) [pure virtual]

Assign a value (abstract). A new value {*val*} is assigned to the vector's position {*pos*}.

Parameters:

pos the position

val the new value

Implemented in ColumnVector, FullVector, ImageVector, and RowVector.

8.56.3.7 virtual int ReferenceVector::size (void) [pure virtual]

Return the vector's size (abstract). Depending on the concrete instance we will get the associated array's number of rows or cols.

Returns:

the vector's size

Implemented in ColumnVector, FullVector, ImageVector, and RowVector.

Referenced by copy().

8.56.3.8 void ReferenceVector::copy (ReferenceVector & *other*) [inline]

Copy another vector's content. If the vectors have different sizes only as much as possible is being copied.

Parameters:

other the other vector

Definition at line 78 of file ReferenceVector.hh.

References at(), and size().

8.56.4 Member Data Documentation

8.56.4.1 int ReferenceVector::m_vroot [protected]

The vector's root. Depending on whether we're a row or column vector (inherited classes) this is either the vector's row or col.

Definition at line 96 of file ReferenceVector.hh.

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

- ReferenceVector.hh

8.57 riff_struct Struct Reference

`#include <avilib.h>`

Public Attributes

- uint8_t id [4]
- uint32_t len
- uint8_t wave_id [4]

8.57.1 Detailed Description

Definition at line 471 of file avilib.h.

8.57.2 Member Data Documentation

8.57.2.1 uint8_t riff_struct::id[4]

Definition at line 473 of file avilib.h.

8.57.2.2 uint32_t riff_struct::len

Definition at line 474 of file avilib.h.

8.57.2.3 uint8_t riff_struct::wave_id[4]

Definition at line 475 of file avilib.h.

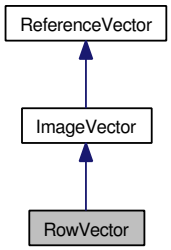
The documentation for this struct was generated from the following file:

- avilib.h

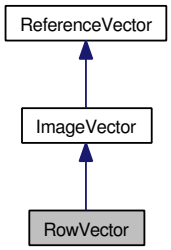
8.58 RowVector Class Reference

`#include <RowVector.hh>`

Inheritance diagram for RowVector:



Collaboration diagram for RowVector:



Public Member Functions

- RowVector (ImageArray< coeff > *ar)
- virtual ~RowVector (void)
- virtual void go (int root)
- virtual coeff at (int pos)
- virtual void to (int pos, coeff val)
- virtual int size (void)
- virtual void update (void)

Private Attributes

- int m_rbase

8.58.1 Detailed Description

Row-reference. A reference to a two-dimensional array's row.

Definition at line 21 of file RowVector.hh.

8.58.2 Constructor & Destructor Documentation

8.58.2.1 RowVector::RowVector (ImageArray< coeff > * *ar*) [inline]

Constructor. As we've got no default constructor we need to call the mother's class' constructor here...

Definition at line 28 of file RowVector.hh.

References go().

8.58.2.2 virtual RowVector::~RowVector (void) [inline, virtual]

Destructor. Does nothing.

Definition at line 30 of file RowVector.hh.

8.58.3 Member Function Documentation

8.58.3.1 virtual void RowVector::go (int *root*) [virtual]

Sets a new root. In this context this "new root" means a new row.

Exceptions:

invalid_argument the new root is greater than the number of rows

Parameters:

root the new row

Reimplemented from ImageVector.

Referenced by RowVector().

8.58.3.2 virtual coeff RowVector::at (int *pos*) [virtual]

Get a value. Return the value at a given column from the current row.

Exceptions:

invalid_argument a negative value was given for the new root

Parameters:

pos the column

Returns:

the value

Implements ImageVector.

8.58.3.3 virtual void RowVector::to (int *pos*, coeff *val*) [virtual]

Assigns a new value. Sets a new value at a given column from the current row.

Parameters:

pos the column

val the new value

Implements ImageVector.

8.58.3.4 virtual int RowVector::size (void) [virtual]

The vector's size. Returns the vector's size (the number of cols)

Returns:

the number of cols

Implements ImageVector.

8.58.3.5 virtual void RowVector::update (void) [virtual]

Updates the vector's settings. This is necessary each time after the array has been resized.

Reimplemented from ImageVector.

8.58.4 Member Data Documentation

8.58.4.1 int RowVector::m_rbase [private]

Internal value. The base to compute the row's entries (row∗xsize)

Definition at line 61 of file RowVector.hh.

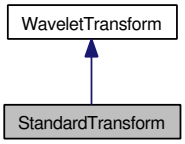
The documentation for this class was generated from the following file:

- RowVector.hh

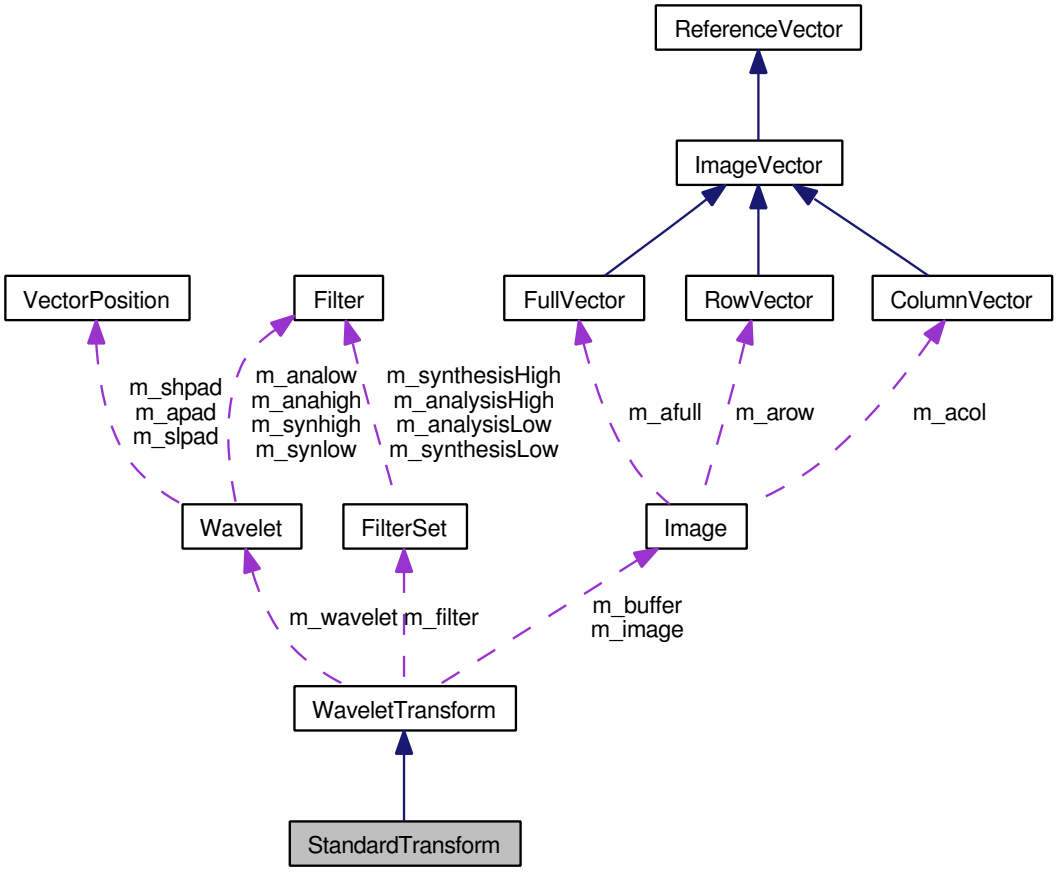
8.59 StandardTransform Class Reference

`#include <StandardTransform.hh>`

Inheritance diagram for StandardTransform:



Collaboration diagram for StandardTransform:



Public Member Functions

- StandardTransform (Image &img, FilterSet &fil)

Protected Member Functions

- virtual void doanalysis (int steps)
- virtual void dosynthesis (int steps, int prevSteps=0)

8.59.1 Detailed Description

A Standard Transform. The transform is two-dimensional, it thus works on images.

Definition at line 22 of file StandardTransform.hh.

8.59.2 Constructor & Destructor Documentation

8.59.2.1 StandardTransform::StandardTransform (Image & *img*, FilterSet & *fil*) [inline]

Constructor. Calls the superclass' constructor.

Parameters:

img the image

fil the filter (wavelet)

Definition at line 31 of file StandardTransform.hh.

8.59.3 Member Function Documentation

8.59.3.1 virtual void StandardTransform::doanalysis (int *steps*) [protected, virtual]

Perform a Standard transform on the image. If a symmetric filter is used, an image can't get fully decomposed. In this case the maximum number of steps is automatically limited to level - 1.

Parameters:

steps the number of transform steps

Implements WaveletTransform.

8.59.3.2 virtual void StandardTransform::dosynthesis (int *steps*, int *prevSteps* = 0) [protected, virtual]

Perform an inverse Standard transform on the image.

Parameters:

steps the number of inverse transform steps

prevSteps if greater than 0, the number of previous decomposition steps to assume. This is necessary to reconstruct images that are not square and have side lengths which are not powers of two.

Implements WaveletTransform.

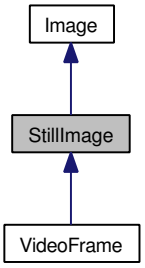
The documentation for this class was generated from the following file:

- StandardTransform.hh

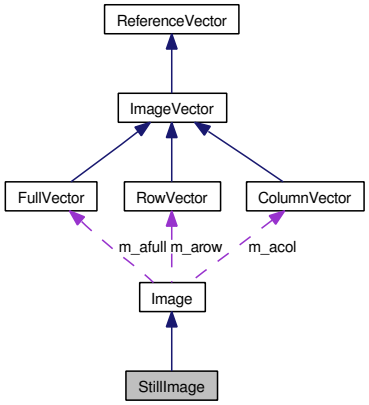
8.60 StillImage Class Reference

`#include <StillImage.hh>`

Inheritance diagram for StillImage:



Collaboration diagram for StillImage:



Public Member Functions

- StillImage (int rows=0, int cols=0)
- StillImage (ImageArray< coeff > &ia)
- virtual ~StillImage (void)
- virtual coeff at (int y, int x) const
- virtual coeff at (int abs) const
- virtual void to (int y, int x, coeff val)
- virtual void to (int abs, coeff val)
- virtual int abs (int y, int x) const
- virtual bool epsilons (Image &si, coeff epsilon) const
- virtual bool equals (Image &si) const
- virtual void read (char const *fname, int rawy=0, int rawx=0)
- virtual void read (char const *fname, filetype ftype, int rawy=0, int rawx=0)

- virtual void write (char const *fname, bool beautify=false)
- virtual void write (char const *fname, filetype ftype, bool beautify=false)
- virtual Image * clone (void) const
- virtual coeff smax (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const
- virtual coeff smin (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const
- virtual coeff amax (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const
- virtual coeff amin (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const
- virtual coeff saverage (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const
- virtual coeff aaverage (int fromY=0, int fromX=0, int toY=-1, int toX=-1) const
- virtual void resize (int rows, int cols)

Static Public Member Functions

- static Image * makeEmpty (int rows=0, int cols=0)

Protected Member Functions

- StillImage (bool dummy)
- virtual Image * mkImage (int rows=0, int cols=0) const

Protected Attributes

- ImageArray< coeff > * m_coeffs
- bool m_coeffsMustDelete

8.60.1 Detailed Description

A still image. A (grey-scale) two-dimensional image.

Definition at line 21 of file StillImage.hh.

8.60.2 Constructor & Destructor Documentation

8.60.2.1 StillImage::StillImage (int *rows* = 0, int *cols* = 0)

Constructor. Creates an empty image with given dimensions.

Parameters:

rows the number of rows

cols the number of cols

Referenced by makeEmpty().

8.60.2.2 StillImage::StillImage (ImageArray< coeff > & *ia*)

Constructor. Imports an existing {*ImageArray*} object. The array will be duplicated.

Parameters:

ia The source array

8.60.2.3 virtual StillImage::~StillImage (void) [virtual]

Destructor. Deallocates objects.

8.60.2.4 StillImage::StillImage (bool *dummy*) [inline, protected]

Dummy constructor, only to be called by derived classes.

Parameters:

dummy only distinguishes this constructor from the others

Definition at line 256 of file StillImage.hh.

8.60.3 Member Function Documentation

8.60.3.1 virtual coeff StillImage::at (int *y*, int *x*) const [inline, virtual]

Get a value. Returns the value at row {*y*} and col {*x*}.

Parameters:

y the row

x the col

Returns:

the value

Implements Image.

Definition at line 47 of file StillImage.hh.

References m_coeffs.

8.60.3.2 virtual coeff StillImage::at (int *abs*) const [inline, virtual]

Get a value. Returns the value at absolute position {*abs*}.

Parameters:

abs the position

Returns:

the value

Implements Image.

Definition at line 53 of file StillImage.hh.

References m_coeffs.

8.60.3.3 virtual void StillImage::to (int *y*, int *x*, coeff *val*) [inline, virtual]

Set a value. Sets the value at row {*y*} and col {*x*}.

Parameters:

y the row

x the col

val the new value

Returns:

the value

Implements Image.

Definition at line 63 of file StillImage.hh.

References m_coeffs.

8.60.3.4 virtual void StillImage::to (int *abs*, coeff *val*) [inline, virtual]

Set a value. Sets the value at absolute position {*abs*}.

Parameters:

abs the position

val the new value

Implements Image.

Definition at line 69 of file StillImage.hh.

References m_coeffs.

8.60.3.5 virtual int StillImage::abs (int *y*, int *x*) const [inline, virtual]

Return the absolute offset of a position.

Parameters:

y the position's row

x the position's col

Returns:

the absolute offset

Implements Image.

Definition at line 78 of file StillImage.hh.

References m_coeffs.

8.60.3.6 virtual bool StillImage::epsilons (Image & *si*, coeff *epsilon*) const [virtual]

Rough comparison. See if two images are similar according to a given {*epsilon}* (important for floating-point comparisons).

Parameters:

si the other {*Image}* object

epsilon the epsilon

Returns:

if both are identical: {*true}*, else {*false}*

Implements Image.

8.60.3.7 virtual bool StillImage::equals (Image & *si*) const [virtual]

Compares two still images. Return {*true}* if both are equal.

Parameters:

si The other {*StillImage}* object

Returns:

if equals: {*true}*, else {*false}*

Implements Image.

8.60.3.8 virtual void StillImage::read (char const * *fname*, int *rawy* = 0, int *rawx* = 0) [virtual]

Read an image. Reads an image from a file guessing the file type from the file name's extension. Currently PGM, RAW and PFI are supported.

Exceptions:

invalid_argument the file type is either not supported or could not be determined from the given file name

Parameters:

fname the file name, if {*NULL*}, then {*stdin*}

rawy the number of rows (only needed for RAW format)

rawx the number of cols (only needed for RAW format)

Implements Image.

8.60.3.9 virtual void StillImage::read (char const * *fname*, filetype *ftype*, int *rawy* = 0, int *rawx* = 0) [virtual]

Read an image. Reads an image from a file using the specified file type. Currently PGM, RAW and PFI are supported.

Exceptions:

invalid_argument the file type is either not supported or could not be determined from the given file name

Parameters:

fname the file name, if {*NULL*}, then {*stdin*}

ftype the file type

rawy the number of rows (only needed for RAW format)

rawx the number of cols (only needed for RAW format)

Implements Image.

8.60.3.10 virtual void StillImage::write (char const * *fname*, bool *beautify* = false) [virtual]

Write an image (abstract). Writes an image to a file guessing the file type from the file name's extension.

Parameters:

fname the file name, if {*NULL*}, then {*stdout*}

beautify beautify images that have not had more analysis than synthesis steps?

Implements Image.

8.60.3.11 virtual void StillImage::write (char const * *fname*, filetype *ftype*, bool *beautify* = false) [virtual]

Write an image (abstract). Writes an image to a file using the specified file type.

Parameters:

ftype the file type

fname the file name, if {*NULL*}, then {*stdout*}

beautify beautify images that have not had more analysis than synthesis steps?

Implements Image.

8.60.3.12 virtual Image* StillImage::clone (void) const [virtual]

Produce a copy. Every dynamically object will be cloned rather than passing on the reference.

Returns:

the new, copied object.

Implements Image.

Reimplemented in VideoFrame.

8.60.3.13 static Image* StillImage::makeEmpty (int *rows* = 0, int *cols* = 0) [inline, static]

Create an empty StillImage.

Parameters:

rows the number of rows

cols the number of cols

Returns:

the new image

Definition at line 156 of file StillImage.hh.

References Image::cols(), Image::rows(), and StillImage().

8.60.3.14 virtual coeff StillImage::smax (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const [inline, virtual]

Returns the maximum value in a region. Signs will be considered.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the maximum

Implements Image.

Definition at line 166 of file StillImage.hh.

References m_coeffs.

8.60.3.15 virtual coeff StillImage::smin (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const [inline, virtual]

Returns the maximum absolute value in a region. Signs will be discarded.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the maximum

Implements Image.

Definition at line 177 of file StillImage.hh.

References m_coeffs.

8.60.3.16 virtual coeff StillImage::amax (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const [inline, virtual]

Returns the maximum absolute value in a region. Signs will be discarded.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the maximum

Implements Image.

Definition at line 188 of file StillImage.hh.

References m_coeffs.

8.60.3.17 virtual coeff StillImage::amin (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const [inline, virtual]

Returns the minimum absolute value in a region. Signs will be discarded.

Parameters:

fromY the first point's row
fromX the first point's col
toY the second point's row (-1 if lower image border)
toX the second point's col (-1 if right image border)

Returns:

the minimum

Implements Image.

Definition at line 199 of file StillImage.hh.

References m_coeffs.

8.60.3.18 virtual coeff StillImage::saverage (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const [inline, virtual]

Return the average color for a rectangular region inside the image drawn from one point within and the second point just outside the region. The coefficients' signedness will be considered.

Parameters:

fromY the first point's row
fromX the first point's col
toY the second point's row (-1 if lower image border)
toX the second point's col (-1 if right image border)

Returns:

the average color

Implements Image.

Definition at line 211 of file StillImage.hh.

References m_coeffs.

8.60.3.19 virtual coeff StillImage::aaverage (int *fromY* = 0, int *fromX* = 0, int *toY* = -1, int *toX* = -1) const [inline, virtual]

Return the average color for a rectangular region inside the image drawn from one point within and the second point just outside the region. The coefficients' signedness will be discarded.

Parameters:

fromY the first point's row

fromX the first point's col

toY the second point's row (-1 if lower image border)

toX the second point's col (-1 if right image border)

Returns:

the average color

Implements Image.

Definition at line 223 of file StillImage.hh.

References m_coeffs.

8.60.3.20 virtual void StillImage::resize (int *rows*, int *cols*) [virtual]

Resize the image's x/y dimensions. The old values will remain as far as they fit in the new dimensions. The new dimensions must all be greater than zero.

Exceptions:

invalid_argument one or both dimensions are either negative or zero

Parameters:

rows the new number of rows

cols the new number of cols

Implements Image.

8.60.3.21 virtual Image* StillImage::mkImage (int *rows* = 0, int *cols* = 0) const [protected, virtual]

Factory method to get a new instance of a given size filled with zeroes.

Parameters:

rows the number of rows

cols the number of cols

Returns:

the new image

Implements Image.

8.60.4 Member Data Documentation

8.60.4.1 ImageArray<coeff>* StillImage::m_coeffs [protected]

The image values. The image values are stored in a two-dimensional array.

Definition at line 259 of file StillImage.hh.

Referenced by aaverage(), abs(), amax(), amin(), at(), saverage(), smax(), smin(), and to().

8.60.4.2 bool StillImage::m_coeffsMustDelete [protected]

This is true if the 'm_coeffs' member variable is private to the object and must therefore be deleted at object destruction

Definition at line 263 of file StillImage.hh.

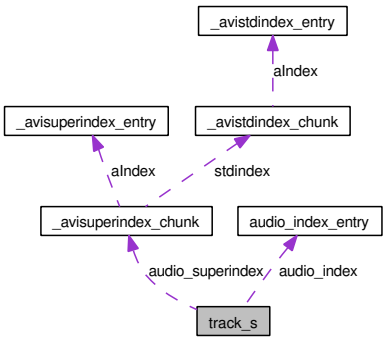
The documentation for this class was generated from the following file:

- StillImage.hh

8.61 track_s Struct Reference

`#include <avilib.h>`

Collaboration diagram for track_s:



Public Attributes

- long a_fmt
- long a_chans
- long a_rate
- long a_bits
- long mp3rate
- long a_vbr
- long padrate
- long audio_strn
- off_t audio_bytes
- long audio_chunks
- char audio_tag [4]
- long audio_posc
- long audio_posb
- off_t a_codech_off
- off_t a_codecf_off
- audio_index_entry * audio_index
- avisuperindex_chunk * audio_superindex

8.61.1 Detailed Description

Definition at line 193 of file avilib.h.

8.61.2 Member Data Documentation

8.61.2.1 long track_s::a_fmt

Definition at line 196 of file avilib.h.

8.61.2.2 long track_s::a_chans

Definition at line 197 of file avilib.h.

8.61.2.3 long track_s::a_rate

Definition at line 198 of file avilib.h.

8.61.2.4 long track_s::a_bits

Definition at line 199 of file avilib.h.

8.61.2.5 long track_s::mp3rate

Definition at line 200 of file avilib.h.

8.61.2.6 long track_s::a_vbr

Definition at line 201 of file avilib.h.

8.61.2.7 long track_s::padrate

Definition at line 202 of file avilib.h.

8.61.2.8 long track_s::audio_strn

Definition at line 204 of file avilib.h.

8.61.2.9 off_t track_s::audio_bytes

Definition at line 205 of file avilib.h.

8.61.2.10 long track_s::audio_chunks

Definition at line 206 of file avilib.h.

8.61.2.11 char track_s::audio_tag[4]

Definition at line 208 of file avilib.h.

8.61.2.12 long track_s::audio_posc

Definition at line 209 of file avilib.h.

8.61.2.13 long track_s::audio_posb

Definition at line 210 of file avilib.h.

8.61.2.14 off_t track_s::a_codech_off

Definition at line 212 of file avilib.h.

8.61.2.15 off_t track_s::a_codecf_off

Definition at line 213 of file avilib.h.

8.61.2.16 audio_index_entry* track_s::audio_index

Definition at line 215 of file avilib.h.

8.61.2.17 avisuperindex_chunk* track_s::audio_superindex

Definition at line 216 of file avilib.h.

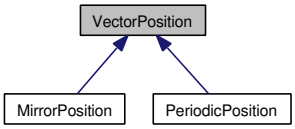
The documentation for this struct was generated from the following file:

- avilib.h

8.62 VectorPosition Class Reference

`#include <VectorPosition.hh>`

Inheritance diagram for VectorPosition:



Public Member Functions

- VectorPosition (void)
- VectorPosition (int newsize)
- virtual ~VectorPosition (void)
- void size (int newsize)
- int size (void)
- int pos (int position)
- int pos (int position, int length)
- int pos (int position, int start, int length)
- virtual int pos (int position, int start, int end, int &sign)=0
- void setsymm (bool symm)
- bool issymm (void)

Protected Attributes

- int m_vsize
- bool m_symmetry

8.62.1 Detailed Description

A Vector index (abstract). Performs symmetric, periodic and whatever extensions to a vector or a part of it.

Definition at line 23 of file VectorPosition.hh.

8.62.2 Constructor & Destructor Documentation

8.62.2.1 VectorPosition::VectorPosition (void)

Constructor. Sets standard values.

8.62.2.2 VectorPosition::VectorPosition (int *newsize*)

Constructor. Sets the real vector size to be used if not passed on to pos () method.

Parameters:

newsize the size

8.62.2.3 virtual VectorPosition::~VectorPosition (void) [inline, virtual]

Destructor. Does nothing.

Definition at line 36 of file VectorPosition.hh.

8.62.3 Member Function Documentation

8.62.3.1 void VectorPosition::size (int *newsize*) [inline]

Set a new size.

Parameters:

newsize the new size

Definition at line 41 of file VectorPosition.hh.

References m_vsize.

8.62.3.2 int VectorPosition::size (void) [inline]

Return the current size.

Returns:

the current size

Definition at line 46 of file VectorPosition.hh.

References m_vsize.

8.62.3.3 int VectorPosition::pos (int *position*) [inline]

Calculate real position. If a position is greater than the size or less than zero we mirror the position back into range.

Parameters:

position the requested position

Returns:

the new position

Definition at line 54 of file VectorPosition.hh.

References m_vsize.

Referenced by pos().

8.62.3.4 int VectorPosition::pos (int *position*, int *length*) [inline]

Calculate real position assuming a given vector length. If a position is greater than the assumed size or less than zero we mirror the position back into range.

Parameters:

position the requested position

length the assumed length

Returns:

the new position

Definition at line 67 of file VectorPosition.hh.

References pos().

8.62.3.5 int VectorPosition::pos (int *position*, int *start*, int *length*) [inline]

Calculate real position assuming a given vector start and length. If a position is greater than the assumed size or less than the assumed start position we mirror the position back into range.

Parameters:

position the requested position

start the assumed start

length the assumed length

Returns:

the new position

Definition at line 82 of file VectorPosition.hh.

References pos().

8.62.3.6 virtual int VectorPosition::pos (int *position*, int *start*, int *end*, int & *sign*) [pure virtual]

Calculate real position assuming a given vector start and length. If a position is greater than the assumed size or less than the assumed start position we mirror the position back into range.

Parameters:

position the requested position
start the assumed start
end the assumed end point
sign set to -1 if a sign change has taken place

Returns:

the new position

Implemented in MirrorPosition, and PeriodicPosition.

8.62.3.7 void VectorPosition::setsymm (bool *symm*) [inline]

Set symmetry property. If set we get a sign change with every half period.

Parameters:

symm if symmetric: {*true*}

Definition at line 105 of file VectorPosition.hh.

References m_symmetry.

8.62.3.8 bool VectorPosition::issymm (void) [inline]

Return the symmetry settings.

Returns:

if symmetric sign change occurs: {*true*} else {*false*}

Definition at line 110 of file VectorPosition.hh.

References m_symmetry.

8.62.4 Member Data Documentation

8.62.4.1 int VectorPosition::m_vsize [protected]

The default vector size.

Definition at line 114 of file VectorPosition.hh.

Referenced by pos(), and size().

8.62.4.2 bool VectorPosition::m_symmetry [protected]

The symmetric sign change setting.

Definition at line 116 of file VectorPosition.hh.

Referenced by issymm(), and setsymm().

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

- VectorPosition.hh

8.63 video_index_entry Struct Reference

`#include <avilib.h>`

Public Attributes

- off_t key
- off_t pos
- off_t len

8.63.1 Detailed Description

Definition at line 119 of file avilib.h.

8.63.2 Member Data Documentation

8.63.2.1 off_t video_index_entry::key

Definition at line 121 of file avilib.h.

8.63.2.2 off_t video_index_entry::pos

Definition at line 122 of file avilib.h.

8.63.2.3 off_t video_index_entry::len

Definition at line 123 of file avilib.h.

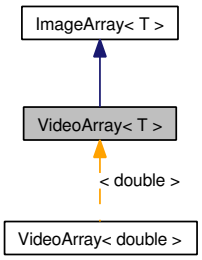
The documentation for this struct was generated from the following file:

- avilib.h

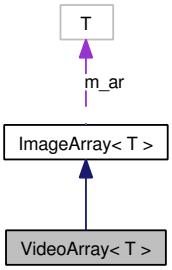
8.64 VideoArray< T > Class Template Reference

`#include <VideoArray.hh>`

Inheritance diagram for VideoArray< T >:



Collaboration diagram for VideoArray< T >:



Public Member Functions

- VideoArray (int rows, int cols, int frames)
- VideoArray (int rows, int cols, int frames, const VideoArray< T > *v)
- virtual ~VideoArray (void)
- void current (int frame)
- int current (void)
- int frames (void) const
- T at (int y, int x) const
- virtual T at (int abs) const
- void to (int y, int x, T val)
- virtual void to (int abs, T val)
- virtual int abs (int y, int x) const
- bool epsilonsFrames (int f1, int f2, T epsilon) const
- bool equalsFrames (int f1, int f2) const
- virtual bool epsilons (ImageArray< T > &ia, T epsilon) const
- virtual void resize (int rows, int cols)
- virtual void reframe (int frames, bool keephead=true)

- virtual void import (int rows, int cols, int frames, T *array)
- virtual void import (int rows, int cols, T *array)
- virtual VideoArray< T > * cloneVideo (void) const
- virtual ImageArray< T > * clone (void) const
- virtual void copy (ImageArray< T > &ia)

Protected Member Functions

- bool epsilonsVideo (VideoArray< T > &va, T epsilon) const

Protected Attributes

- int m_frames
- int m_xyzsize
- int m_current
- int m_offset

8.64.1 Detailed Description

template<class T> class VideoArray< T >

A two-dimensional array wrapper. This allows us to access the video's pixels or coefficients and provides some utility methods. It is the main working horse for the Video framework.

Definition at line 23 of file VideoArray.hh.

8.64.2 Constructor & Destructor Documentation

8.64.2.1 template<class T> VideoArray< T >::VideoArray (int *rows*, int *cols*, int *frames*)

Constructor. Sets the dimension and allocates space. A size of zero is allowed, negative values not.

Parameters:

rows the vertical size

cols the horizontal size

frames the number of frames

8.64.2.2 template<class T> VideoArray< T >::VideoArray (int *rows*, int *cols*, int *frames*, const VideoArray< T > * *v*)

8.64.2.3 template<class T> virtual VideoArray< T >::~VideoArray (void) [virtual]

Destructor. Releases allocated memory.

8.64.3 Member Function Documentation

8.64.3.1 template<class T> void VideoArray< T >::current (int *frame*)

Sets the current frame. Valid values are 0..n-1.

Parameters:

frame the current frame

Exceptions:

invalid_argument the new frame is out of bounds

8.64.3.2 template<class T> int VideoArray< T >::current (void) [inline]

Returns the number of the current frame. Counting starts from zero.

Returns:

the number of the current frame.

Definition at line 49 of file VideoArray.hh.

8.64.3.3 template<class T> int VideoArray< T >::frames (void) const [inline]

Returns the number of frames

Returns:

the number of frames

Definition at line 54 of file VideoArray.hh.

8.64.3.4 template<class T> T VideoArray< T >::at (int *y*, int *x*) const [inline]

Returns the value at (x,y).

Parameters:

y the row

x the col

Returns:

the value

Reimplemented from ImageArray< T >.

Definition at line 63 of file VideoArray.hh.

8.64.3.5 template<class T> virtual T VideoArray< T >::at (int *abs*) const [inline, virtual]

Returns the current frame's {*n*}*th* value.

Parameters:

abs the offset from the array start

Returns:

the value

Reimplemented from ImageArray< T >.

Definition at line 69 of file VideoArray.hh.

8.64.3.6 template<class T> void VideoArray< T >::to (int *y*, int *x*, T *val*) [inline]

Sets the value at (x,y).

Parameters:

y the row

x the col

val the value

Reimplemented from ImageArray< T >.

Definition at line 79 of file VideoArray.hh.

8.64.3.7 template<class T> virtual void VideoArray< T >::to (int *abs*, T *val*) [inline, virtual]

Sets the array's {*n*}*th* value.

Parameters:

abs the offset from the array start

val the value

Reimplemented from ImageArray< T >.

Definition at line 85 of file VideoArray.hh.

8.64.3.8 template<class T> virtual int VideoArray< T >::abs (int *y*, int *x*) const [inline, virtual]

Return the absolute offset of a position.

Parameters:

y the position's row

x the position's col

Returns:

the absolute offset

Reimplemented from ImageArray< T >.

Definition at line 93 of file VideoArray.hh.

Referenced by VideoArray< double >::at(), and VideoArray< double >::to().

8.64.3.9 template<class T> bool VideoArray< T >::epsilonsFrames (int *f1*, int *f2*, T *epsilon*) const

Rough comparison. See if this and another frame are similar according to a given {*epsilon*} (important for floating-point comparisons).

Exceptions:

invalid_argument one of the frames is out of bounds

Parameters:

f1 the first frame

f2 the second frame

epsilon the epsilon

Returns:

if both are identical: {*true*}, else {*false*}

8.64.3.10 template<class T> bool VideoArray< T >::equalsFrames (int *f1*, int *f2*) const

Exact comparison. See if two frames are similar.

Exceptions:

invalid_argument one of the frames is out of bounds

Parameters:

f1 the first frame

f2 the second frame if both are identical: {*true*}, else {*false*}

8.64.3.11 template<class T> virtual bool VideoArray< T >::epsilons (ImageArray< T > & *ia*, T *epsilon*) const [virtual]

Rough comparison. See if this and another video or the current and another frame are similar according to a given {*epsilon*} (important for floating-point comparisons). Whether the argument will be treated as a video or a frame depends on its type.

Parameters:

ia the other {*ImageArray*} object

epsilon the epsilon

Returns:

if both are identical: {*true*}, else {*false*}

Reimplemented from ImageArray< T >.

8.64.3.12 template<class T> virtual void VideoArray< T >::resize (int *rows*, int *cols*) [virtual]

Resize the array. The old values will be copied to the new dimension as far as they fit in. The new dimensions must all be greater than zero.

Exceptions:

invalid_argument one or both dimensions are either negative or zero

Parameters:

rows the new number of rows

cols the new number of cols

Reimplemented from ImageArray< T >.

8.64.3.13 template<class T> virtual void VideoArray< T >::reframe (int *frames*, bool *keephead* = true) [virtual]

Sets new number of frames. The old frames will be copied to the new ones as far as they fit in. The new number must be greater than zero.

Exceptions:

invalid_argument the new number of frames is either negative or zero

Parameters:

frames the new number of frames.

keephead if shrinking the video, frames will be deleted at the end.

8.64.3.14 template<class T> virtual void VideoArray< T >::import (int *rows*, int *cols*, int *frames*, T * *array*) [virtual]

Import a raw array. The new dimensions and the new array will be stored discarding the old ones.

Parameters:

rows the new number of rows

cols the new number of cols

frames the new number of frames

array the new array

8.64.3.15 template<class T> virtual void VideoArray< T >::import (int *rows*, int *cols*, T * *array*) [virtual]

This method cannot be used on videos and will throw an invalid_argument.

Parameters:

rows the new number of rows

cols the new number of cols

array the new array

Exceptions:

invalid_argument always

Reimplemented from ImageArray< T >.

8.64.3.16 template<class T> virtual VideoArray<T>* VideoArray< T >::cloneVideo (void) const [virtual]

Create a copy of the whole video. All frames will be duplicated rather than the references.

Returns:

the new object

8.64.3.17 template<class T> virtual ImageArray<T>* VideoArray< T >::clone (void) const [virtual]

Create a copy of the current frame. All frames will be duplicated rather than the references.

Returns:

the new object

Reimplemented from ImageArray< T >.

8.64.3.18 template<class T> virtual void VideoArray< T >::copy (ImageArray< T > & *ia*) [virtual]

Copy from a different array. This includes resizing and reframing if necessary.

Exceptions:

invalid_argument if the other object is not a VideoArray.

Parameters:

ia the other array

Reimplemented from ImageArray< T >.

8.64.3.19 template<class T> bool VideoArray< T >::epsilonsVideo (VideoArray< T > & *va*, T *epsilon*) const [protected]

Rough comparison. See if this and another video are similar according to a given {*epsilon*} (important for floating-point comparisons). Called by 'epsilons()'.

Parameters:

va the other {*ImageArray*} object

epsilon the epsilon

Returns:

if both are identical: {*true*}, else {*false*}

8.64.4 Member Data Documentation

8.64.4.1 template<class T> int VideoArray< T >::m_frames [protected]

The video's number of frames.

Definition at line 192 of file VideoArray.hh.

Referenced by VideoArray< double >::frames().

8.64.4.2 template<class T> int VideoArray< T >::m_xyzsize [protected]

The video's total size.

Definition at line 194 of file VideoArray.hh.

8.64.4.3 template<class T> int VideoArray< T >::m_current [protected]

The video's current frame number.

Definition at line 196 of file VideoArray.hh.

Referenced by VideoArray< double >::current().

8.64.4.4 template<class T> int VideoArray< T >::m_offset [protected]

The offset in the array to the current frame.

Definition at line 198 of file VideoArray.hh.

Referenced by VideoArray< double >::abs(), VideoArray< double >::at(), and VideoArray< double >::to().

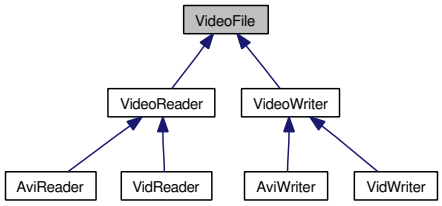
The documentation for this class was generated from the following file:

- VideoArray.hh

8.65 VideoFile Class Reference

`#include <VideoFile.hh>`

Inheritance diagram for VideoFile:



Public Member Functions

- VideoFile (char const *name, VideoArray< coeff > *arrays[ ], int channels, int frames)
- virtual ~VideoFile (void)
- clrmodel colormodel (void)
- void colormodel (clrmodel cm)
- int channels (void) const
- int frames (void) const
- clrmodel colormodel (void) const

Protected Member Functions

- void init (VideoArray< coeff > *videos[ ])

Protected Attributes

- VideoArray< coeff > ** m_arrays
- int m_channels
- char const * m_fname
- clrmodel m_cmodel
- int m_frames

8.65.1 Detailed Description

An abstract image file. A framework to create readers and writers on any color images.

Definition at line 24 of file VideoFile.hh.

8.65.2 Constructor & Destructor Documentation

8.65.2.1 VideoFile::VideoFile (char const * *name*, VideoArray< coeff > * *arrays*[ ], int *channels*, int *frames*)

Constructor. Initializes internal fields and gets an {*ImageArray*} object that may already contain an image or will get one later.

Parameters:

name the file name

arrays the array of video arrays for the values

channels the number of colors

frames the number of frames

8.65.2.2 virtual VideoFile::~VideoFile (void) [virtual]

Destructor. Releases some memory.

8.65.3 Member Function Documentation

8.65.3.1 clrmodel VideoFile::colormodel (void) [inline]

Returns the current color model.

Returns:

the current color model

Definition at line 46 of file VideoFile.hh.

References m_cmodel.

8.65.3.2 void VideoFile::colormodel (clrmodel *cm*) [inline]

Sets a new color model.

Parameters:

cm the new color model

Definition at line 49 of file VideoFile.hh.

References m_cmodel.

8.65.3.3 int VideoFile::channels (void) const [inline]

Return the number of channels.

Returns:

the number of channels

Definition at line 53 of file VideoFile.hh.

References m_channels.

8.65.3.4 int VideoFile::frames (void) const [inline]

Return the number of frames.

Returns:

the number of frames

Definition at line 57 of file VideoFile.hh.

References m_frames.

8.65.3.5 clrmodel VideoFile::colormodel (void) const [inline]

Return the color model.

Returns:

the color model

Definition at line 61 of file VideoFile.hh.

References m_cmodel.

8.65.3.6 void VideoFile::init (VideoArray< coeff > * *videos*[ ]) [protected]

internal initialization

8.65.4 Member Data Documentation

8.65.4.1 VideoArray< coeff >** VideoFile::m_arrays [protected]

The color channels.

Definition at line 66 of file VideoFile.hh.

8.65.4.2 int VideoFile::m_channels [protected]

The number of channels (usually: colors).

Definition at line 68 of file VideoFile.hh.

Referenced by channels().

8.65.4.3 char const* VideoFile::m_fname [protected]

The file name. The name of the file associated with this object.

Definition at line 70 of file VideoFile.hh.

8.65.4.4 clrmodel VideoFile::m_cmodel [protected]

The way the current image's colors are encoded.

Definition at line 72 of file VideoFile.hh.

Referenced by colormodel().

8.65.4.5 int VideoFile::m_frames [protected]

The number of frames

Definition at line 74 of file VideoFile.hh.

Referenced by frames().

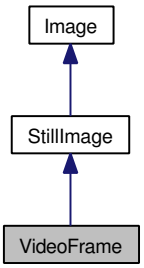
The documentation for this class was generated from the following file:

- VideoFile.hh

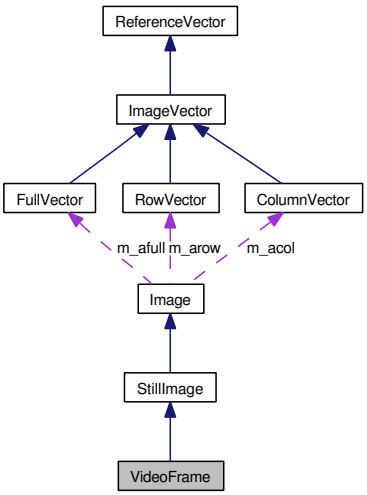
8.66 VideoFrame Class Reference

`#include <VideoFrame.hh>`

Inheritance diagram for VideoFrame:



Collaboration diagram for VideoFrame:



Public Member Functions

- VideoFrame (VideoArray< coeff > *va)
- virtual ~VideoFrame (void)
- virtual Image * clone (void) const

8.66.1 Detailed Description

A video frame in black-white. Almost identical to StillImage, contains some optimization stuff for videos.

Definition at line 23 of file VideoFrame.hh.

8.66.2 Constructor & Destructor Documentation

8.66.2.1 VideoFrame::VideoFrame (VideoArray< coeff > * *va*)

Constructor. Creates an empty image with given dimensions. Unlike in the StillImage class the memory holding the coefficients is held externally and contains all frames. The VideoArray object automatically returns data from the frame which has previously been set as the current one. No boundary checking is done!

Parameters:

va a pointer to the VideoArray holding all frames

8.66.2.2 virtual VideoFrame::~VideoFrame (void) [virtual]

Destructor. Deallocates objects.

8.66.3 Member Function Documentation

8.66.3.1 virtual Image* VideoFrame::clone (void) const [virtual]

Produce a copy. Every dynamically object will be cloned rather than passing on the reference. This operation returns a StillImage and not a VideoFrame, since a VideoFrame always depends on the external VideoArray, so that the copy would only alias to the same external memory!

Returns:

the new, copied object.

Reimplemented from StillImage.

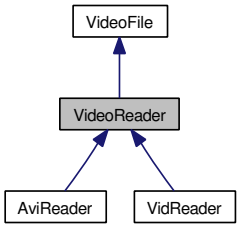
The documentation for this class was generated from the following file:

- VideoFrame.hh

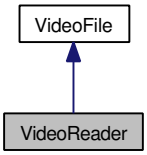
8.67 VideoReader Class Reference

`#include <VideoReader.hh>`

Inheritance diagram for VideoReader:



Collaboration diagram for VideoReader:



Public Member Functions

- VideoReader (char const *name, VideoArray< coeff > *arrays[ ], int channels, int from, int to)
- virtual ~VideoReader (void)
- virtual void read (void)

Protected Member Functions

- virtual int readfmt (void)=0

Protected Attributes

- int m_to
- int m_from

8.67.1 Detailed Description

An abstract color video reader. The video is being read putting its color channels into separate greyscale video objects.

Definition at line 24 of file VideoReader.hh.

8.67.2 Constructor & Destructor Documentation

8.67.2.1 VideoReader::VideoReader (char const * *name*, VideoArray< coeff > * *arrays*[ ], int *channels*, int *from*, int *to*) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference.

Parameters:

name the file name
arrays the array of video arrays for the values
channels the number of colors
from the first frame
to the last frame

Definition at line 39 of file VideoReader.hh.

References m_from, and m_to.

8.67.2.2 virtual VideoReader::~VideoReader (void) [inline, virtual]

Destructor - does nothing.

Definition at line 44 of file VideoReader.hh.

8.67.3 Member Function Documentation

8.67.3.1 virtual void VideoReader::read (void) [virtual]

Read the video . All steps independent of the file format will be performed, like testing for file readability etc.

Exceptions:

invalid_argument invalid file format
ios_base::failure a read error has occured [not supported by all libraries, so eventually {*invalid_argument*} instead]

8.67.3.2 virtual int VideoReader::readfmt (void) [protected, pure virtual]

Read different file formats (abstract). This is the method to be implemented for every video file format. It will be called by the {*read* ()} method.

Returns:

0 if successful, -1 on read error, -2 on file format error.

Implemented in AviReader, and VidReader.

8.67.4 Member Data Documentation

8.67.4.1 int VideoReader::m_to [protected]

the last frame

Definition at line 64 of file VideoReader.hh.

Referenced by VideoReader().

8.67.4.2 int VideoReader::m_from [protected]

the first frame

Definition at line 66 of file VideoReader.hh.

Referenced by VideoReader().

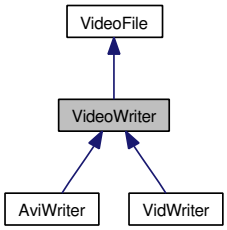
The documentation for this class was generated from the following file:

- VideoReader.hh

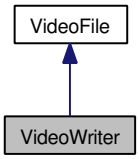
8.68 VideoWriter Class Reference

`#include <VideoWriter.hh>`

Inheritance diagram for VideoWriter:



Collaboration diagram for VideoWriter:



Public Member Functions

- VideoWriter (char const *name, VideoArray< coeff > *arrays[ ], int colors)
- virtual ~VideoWriter (void)
- void write (void)

Protected Member Functions

- virtual int writefmt (void)=0

8.68.1 Detailed Description

An abstract video writer. The video is being written putting its content from an array of greyscale video objects into the file.

Definition at line 23 of file VideoWriter.hh.

8.68.2 Constructor & Destructor Documentation

8.68.2.1 VideoWriter::VideoWriter (char const * *name*, VideoArray< coeff > * *arrays*[ ], int *colors*) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference.

Parameters:

name the file name

arrays the array of video arrays for the values

colors the number of colors

Definition at line 36 of file VideoWriter.hh.

8.68.2.2 virtual VideoWriter::~VideoWriter (void) [inline, virtual]

Destructor - does nothing.

Definition at line 40 of file VideoWriter.hh.

8.68.3 Member Function Documentation

8.68.3.1 void VideoWriter::write (void)

Write the video . All steps independent of the file format will be performed, like testing for file writeability etc.

Exceptions:

ios_base::failure a read error has occured [not supported by all libraries, so eventually {*invalid_argument*} instead]

8.68.3.2 virtual int VideoWriter::writefmt (void) [protected, pure virtual]

Write different file formats (abstract). This is the method to be implemented for every video file format. It will be called by the {*write* ()} method.

Returns:

0 if successful, -1 on write error.

Implemented in AviWriter, and VidWriter.

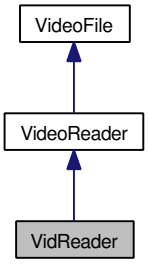
The documentation for this class was generated from the following file:

- VideoWriter.hh

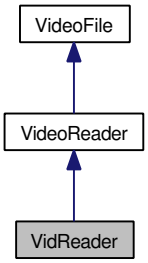
8.69 VidReader Class Reference

`#include <VidReader.hh>`

Inheritance diagram for VidReader:



Collaboration diagram for VidReader:



Public Member Functions

- VidReader (char const *name, VideoArray< coeff > *arrays[ ], int rawy, int rawx, int colors, int from, int to, int skip=0)
- virtual ~VidReader (void)

Static Public Member Functions

- static int framesInFile (const char *fname, int rows, int cols, int colors, int skip)

Protected Member Functions

- virtual int readfmt (void)

Protected Attributes

- int m_ysize

- int m_xsize
- int m_skip

8.69.1 Detailed Description

A VID file reader.

Definition at line 21 of file VidReader.hh.

8.69.2 Constructor & Destructor Documentation

8.69.2.1 VidReader::VidReader (char const * *name*, VideoArray< coeff > * *arrays*[ ], int *rawy*, int *rawx*, int *colors*, int *from*, int *to*, int *skip* = 0)

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference.

Parameters:

name the file name

arrays the array of video arrays for the values

rawy the number of rows

rawx the number of cols

colors the number of colors

from the first frame

to one greater than the last frame, if equal to 'from' then the whole video will be read, if 0, then the video will be read from 'from' to the end

skip the size of the header to skip

8.69.2.2 virtual VidReader::~VidReader (void) [inline, virtual]

Destructor - does nothing.

Definition at line 43 of file VidReader.hh.

8.69.3 Member Function Documentation

8.69.3.1 static int VidReader::framesInFile (const char * *fname*, int *rows*, int *cols*, int *colors*, int *skip*) [static]

Return the number of frames in a file.

Parameters:

fname the file name.

rows the number of rows

cols the number of columns

colors the number of colors

skip the number bytes to skip

Returns:

the number of frames

8.69.3.2 virtual int VidReader::readfmt (void) [protected, virtual]

Read the VID file format. This does the actual work of reading and parsing the video file. It gets called by the {*read* ()} method.

Returns:

0 if successful, -1 on read error, -2 on file format error.

Implements VideoReader.

8.69.4 Member Data Documentation

8.69.4.1 int VidReader::m_ysize [protected]

The assumed number of rows

Definition at line 62 of file VidReader.hh.

8.69.4.2 int VidReader::m_xsize [protected]

The assumed number of columns

Definition at line 64 of file VidReader.hh.

8.69.4.3 int VidReader::m_skip [protected]

If applicable, a header to skip before reading the contents

Definition at line 66 of file VidReader.hh.

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

- VidReader.hh

8.70 VidWriter Class Reference

`#include <VidWriter.hh>`

Inheritance diagram for VidWriter:

Collaboration diagram for VidWriter:

Public Member Functions

- VidWriter (char const *name, VideoArray< coeff > *arrays[ ], int colors, int skip)
- virtual ~VidWriter (void)

Protected Member Functions

- virtual int writefmt (void)

Protected Attributes

- int m_skip

8.70.1 Detailed Description

A VID video file writer. Dimensions are known from the {*Image*} object.

Definition at line 22 of file VidWriter.hh.

8.70.2 Constructor & Destructor Documentation

8.70.2.1 VidWriter::VidWriter (char const * *name*, VideoArray< coeff > * *arrays*[ ], int *colors*, int *skip*) [inline]

Constructor. Only calls the mother class' constructor to initialize the {*ImageArray*} reference and set the offset if necessary.

Parameters:

name the file name

arrays the array of video arrays for the values

colors the number of colors

skip the size of the header to fill with zeros before the actual contents

Definition at line 36 of file VidWriter.hh.

References m_skip.

8.70.2.2 virtual VidWriter::~VidWriter (void) [inline, virtual]

Destructor - does nothing.

Definition at line 41 of file VidWriter.hh.

8.70.3 Member Function Documentation

8.70.3.1 virtual int VidWriter::writefmt (void) [protected, virtual]

Write a VID video file. This is the method that does the actual work. It will be called by the {*write* ()} method.

Returns:

0 if successful, -1 on write error.

Implements VideoWriter.

8.70.4 Member Data Documentation

8.70.4.1 int VidWriter::m_skip [protected]

The number of bytes before the contents

Definition at line 50 of file VidWriter.hh.

Referenced by VidWriter().

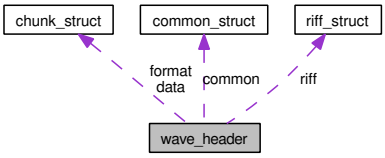
The documentation for this class was generated from the following file:

- VidWriter.hh

8.71 wave_header Struct Reference

`#include <avilib.h>`

Collaboration diagram for wave_header:



Public Attributes

- struct riff_struct riff
- struct chunk_struct format
- struct common_struct common
- struct chunk_struct data

8.71.1 Detailed Description

Definition at line 495 of file avilib.h.

8.71.2 Member Data Documentation

8.71.2.1 struct riff_struct wave_header::riff [read]

Definition at line 497 of file avilib.h.

8.71.2.2 struct chunk_struct wave_header::format [read]

Definition at line 498 of file avilib.h.

8.71.2.3 struct common_struct wave_header::common [read]

Definition at line 499 of file avilib.h.

8.71.2.4 struct chunk_struct wave_header::data [read]

Definition at line 500 of file avilib.h.

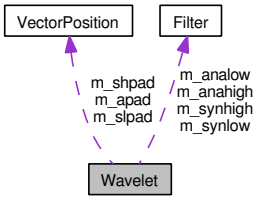
The documentation for this struct was generated from the following file:

- avilib.h

8.72 Wavelet Class Reference

`#include <Wavelet.hh>`

Collaboration diagram for Wavelet:



Public Member Functions

- int level (ReferenceVector &dest, ReferenceVector &src, int steps)
- Wavelet (FilterSet &fset)
- virtual ~Wavelet (void)
- void analysis (ReferenceVector &dest, ReferenceVector &src, int steps)
- void synthesis (ReferenceVector &dest, ReferenceVector &src, int steps, int prevSteps=0)
- virtual void anastep (ReferenceVector &dest, ReferenceVector &src, int size)
- virtual void synstep (ReferenceVector &dest, ReferenceVector &src, int size)

Protected Member Functions

- void init (FilterSet *pfset=NULL)

Protected Attributes

- Filter * m_analow
- Filter * m_anahigh
- Filter * m_synlow
- Filter * m_synhigh
- VectorPosition * m_apad
- VectorPosition * m_slpad
- VectorPosition * m_shpad
- bool m_symmetric
- int m_npad
- int * m_apositions
- int m_apsize
- int * m_slpositions
- int m_slpsize
- int * m_shpositions
- int * m_shsigns
- int m_shpsize

8.72.1 Detailed Description

A Wavelet. The Wavelet transform is one-dimensional, it thus works on vectors.

Definition at line 24 of file Wavelet.hh.

8.72.2 Constructor & Destructor Documentation

8.72.2.1 Wavelet::Wavelet (FilterSet & *fset*)

Constructor. Sets the the filter.

8.72.2.2 virtual Wavelet::~Wavelet (void) [virtual]

Destructor. Releases allocated objects.

8.72.3 Member Function Documentation

8.72.3.1 int Wavelet::level (ReferenceVector & *dest*, ReferenceVector & *src*, int *steps*)

Calculate the level for the transform and execute some consistency checks.

Exceptions:

invalid_argument vector size is not a power of two or the two vectors don't have the same size

Parameters:

dest the destination vector

src the source vector

steps the number of steps for the transform

8.72.3.2 void Wavelet::analysis (ReferenceVector & *dest*, ReferenceVector & *src*, int *steps*)

Decompose a vector. The result is written to a second vector.

Exceptions:

invalid_argument low pass subband signal too small (decrease the number of transform steps or increase signal size)

Parameters:

src A vector of {*coeff*} containing the original data

dest A vector of {*coeff}* to get the transformed data

steps The last col the decomposition step will be performed to

8.72.3.3 void Wavelet::synthesis (ReferenceVector & *dest*, ReferenceVector & *src*, int *steps*, int *prevSteps* = 0)

Reconstruct a vector. The result is written to a second vector.

Parameters:

src A vector of {*coeff}* containing the analysisd data

dest A vector of {*coeff}* to get the synthesised data

steps The number of coefficient pairs to synthesis.

prevSteps if greater than 0, the number of previous decomposition steps to assume. This is necessary to reconstruct images that are not square and have side lengths which are not powers of two.

8.72.3.4 virtual void Wavelet::anastep (ReferenceVector & *dest*, ReferenceVector & *src*, int *size*) [virtual]

A decomposition step on a vector. The result is written to a second vector. Only the result coefficients are being written, the rest may have to be done manually.

Parameters:

dest A vector of {*coeff}* to get the transformed data

src A vector of {*coeff}* containing the original data

size The number of values in {*src}* to perform the decomposition step on

8.72.3.5 virtual void Wavelet::synstep (ReferenceVector & *dest*, ReferenceVector & *src*, int *size*) [virtual]

A reconstruction step on a vector. The result is written to a second vector. Only the result coefficients are being written, the rest may have to be done manually.

Parameters:

dest A vector of {*coeff}* to get the transformed data

src A vector of {*coeff}* containing the original data

size The number of values in {*src}* to perform the synthesis step on

8.72.3.6 void Wavelet::init (FilterSet * *pfset* = NULL) [protected]

Initialize the filter set to be used. The filters in the set will be duplicated.

Parameters:

pfset a reference to the filter set

8.72.4 Member Data Documentation

8.72.4.1 Filter* Wavelet::m_analow [protected]

The low pass analysis filter.

Definition at line 111 of file Wavelet.hh.

8.72.4.2 Filter* Wavelet::m_anahigh [protected]

The high pass analysis filter.

Definition at line 113 of file Wavelet.hh.

8.72.4.3 Filter* Wavelet::m_synlow [protected]

The low pass synthesis filter.

Definition at line 115 of file Wavelet.hh.

8.72.4.4 Filter* Wavelet::m_synhigh [protected]

The high pass synthesis filter.

Definition at line 117 of file Wavelet.hh.

8.72.4.5 VectorPosition* Wavelet::m_apad [protected]

The padding object for the analysis.

Definition at line 119 of file Wavelet.hh.

8.72.4.6 VectorPosition* Wavelet::m_slpad [protected]

The padding object for the synthesis low pass part.

Definition at line 121 of file Wavelet.hh.

8.72.4.7 VectorPosition* Wavelet::m_shpad [protected]

The padding object for the synthesis high pass part.

Definition at line 123 of file Wavelet.hh.

8.72.4.8 bool Wavelet::m_symmetric [protected]

{*true}* if filter set is symmetric.

Definition at line 125 of file Wavelet.hh.

8.72.4.9 int Wavelet::m_npad [protected]

The number of padding coefficients needed.

Definition at line 127 of file Wavelet.hh.

8.72.4.10 int* Wavelet::m_apositions [protected]

Padding positions for analysis.

Definition at line 129 of file Wavelet.hh.

8.72.4.11 int Wavelet::m_apsize [protected]

Size of padding positions array for analysis.

Definition at line 131 of file Wavelet.hh.

8.72.4.12 int* Wavelet::m_slpositions [protected]

Padding positions for lowband synthesis.

Definition at line 133 of file Wavelet.hh.

8.72.4.13 int Wavelet::m_slpsize [protected]

Size of padding positions array for lowband synthesis .

Definition at line 135 of file Wavelet.hh.

8.72.4.14 int* Wavelet::m_shpositions [protected]

Padding positions for highband synthesis.

Definition at line 137 of file Wavelet.hh.

8.72.4.15 int* Wavelet::m_shsigns [protected]

Padding signs for highband synthesis.

Definition at line 139 of file Wavelet.hh.

8.72.4.16 int Wavelet::m_shpsize [protected]

Size of padding positions array for highband synthesis .

Definition at line 141 of file Wavelet.hh.

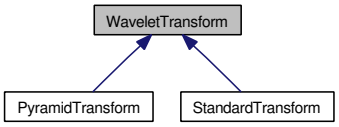
The documentation for this class was generated from the following file:

- Wavelet.hh

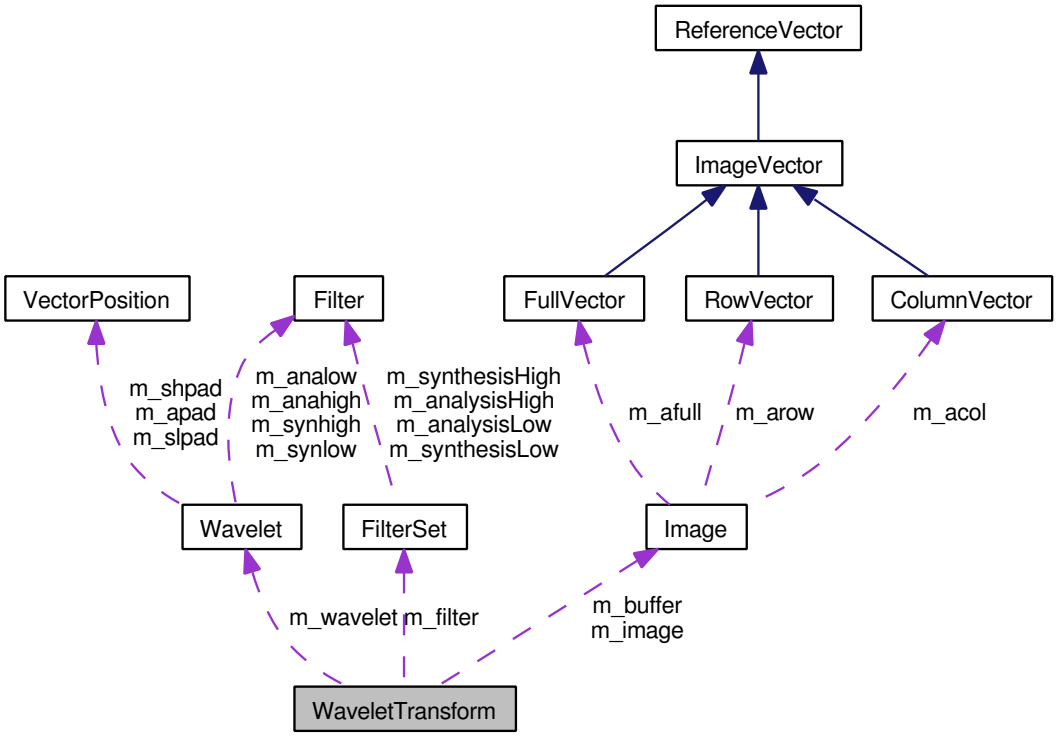
8.73 WaveletTransform Class Reference

`#include <WaveletTransform.hh>`

Inheritance diagram for WaveletTransform:



Collaboration diagram for WaveletTransform:



Public Member Functions

- WaveletTransform (Image &img, FilterSet &fil)
- virtual ~WaveletTransform (void)
- virtual void analysis (int steps)
- virtual void synthesis (int steps, int prevSteps=0)
- void expandImage (void)
- void restoreImage (void)

- virtual void where (area what, int subband, int &yoffs, int &xoffs, int &ysize, int &xsize) const
- virtual void where (int ypos, int xpos, int &subband, area &channel, int steps=-1) const
- virtual void mapPosition (int yOld, int xOld, int toSubband, area toChannel, int &yNew, int &xNew, int &sizeFactor, int steps=-1) const
- virtual coeff saverage (int subband, area channel) const
- virtual coeff aaverage (int subband, area channel) const
- virtual coeff sqvariance (int subband, area channel) const
- virtual coeff variance (int subband, area channel, bool abs=false) const
- virtual coeff sdeviation (int subband, area channel, bool abs=false) const
- virtual coeff ratio (int subband1, int subband2)
- Image * ll (int steps=-1)
- Image * hl (int steps=-1)
- Image * lh (int steps=-1)
- Image * hh (int steps=-1)
- Image * highMax (int steps=-1)
- Image & image (void) const
- virtual Image * subband (area what, int steps=-1)
- virtual int getSubband (int ypos, int xpos, int steps=-1)
- int getSubband (CoeffInformation &c, int steps=-1)
- virtual area getArea (int ypos, int xpos, int steps=-1)
- area getArea (CoeffInformation &c, int steps=-1)
- virtual void import (Image &img, area what, int steps=-1)
- virtual void fill (coeff value, area what, int steps=-1)
- int steps (void) const

Protected Member Functions

- virtual void doanalysis (int steps)=0
- virtual void dosynthesis (int steps, int prevSteps=0)=0

Protected Attributes

- Wavelet * m_wavelet
- Image * m_image
- Image * m_buffer
- FilterSet * m_filter
- int m_rows
- int m_cols

Private Member Functions

- void sanity (void)

8.73.1 Detailed Description

An abstract Wavelet Transform. The Transform transform is two-dimensional, it thus works on images.

Definition at line 26 of file WaveletTransform.hh.

8.73.2 Constructor & Destructor Documentation

8.73.2.1 WaveletTransform::WaveletTransform (Image & *img*, FilterSet & *fil*)

Constructor. Initializes references to Image and Wavelet.

Parameters:

img the image

fil the filter set for the wavelet

8.73.2.2 virtual WaveletTransform::~WaveletTransform (void) [virtual]

Destructor. Releases allocated objects.

8.73.3 Member Function Documentation

8.73.3.1 virtual void WaveletTransform::analysis (int *steps*) [virtual]

Perform a Wavelet transform on the image.

Exceptions:

invalid_argument image is not square

Parameters:

steps the number of transform steps

8.73.3.2 virtual void WaveletTransform::synthesis (int *steps*, int *prevSteps* = 0) [virtual]

Perform an inverse Wavelet transform on the image.

Exceptions:

invalid_argument image is not square

Parameters:

steps the number of inverse transform steps

prevSteps if greater than 0, the number of previous decomposition steps to assume. This is necessary to reconstruct images that are not square and have side lengths which are not powers of two.

8.73.3.3 void WaveletTransform::expandImage (void)

Temporarily expand the image (square, side length a power of two) to make a transform possible.

8.73.3.4 void WaveletTransform::restoreImage (void)

Restore the temporarily expanded image back to original size.

8.73.3.5 virtual void WaveletTransform::where (area *what*, int *subband*, int & *yoffs*, int & *xoffs*, int & *ysize*, int & *xsize*) const [virtual]

Identify position and size of a given channel in a given subband. This default method of calculation refers to the Pyramid transform and due to the lack of alternatives also to the Standard transform. Decomposition schemes using a different geometries, like e.g. the Packet transform may have to implement their own version of that method.

Parameters:

what the channel we want

subband the subband we are looking at

yoffs the returned row

xoffs the returned col

ysize the returned number of rows

xsize the returned number of cols

8.73.3.6 virtual void WaveletTransform::where (int *ypos*, int *xpos*, int & *subband*, area & *channel*, int *steps* = -1) const [virtual]

Identify channel and subband of a given position in the transformed image. This method depends on the 'where' method and thus uses the geometry associated with the 'Pyramid' decomposition.

Parameters:

ypos the location's row.

xpos the location's column.

subband where the calculated subband gets written to.

channel where the calculated channel gets written to.

steps the current number of transform steps to be assumed (-1 for automatic determination)

8.73.3.7 virtual void WaveletTransform::mapPosition (int *yOld*, int *xOld*, int *toSubband*, area *toChannel*, int & *yNew*, int & *xNew*, int & *sizeFactor*, int *steps* = −1) const [virtual]

Identify channel and subband of a given position in the transformed image and map it to another subband/channel. This default method of calculation refers to the Pyramid transform and due to the lack of alternatives also to the Standard transform. Decomposition schemes using a different geometries, like e.g. the Packet transform may have to implement their own version of that method.

Parameters:

yOld the source location's row.

xOld the source location's column.

toSubband the target subband

toChannel the target channel

yNew where the target location's row gets written to.

xNew where the target location's column gets written to.

sizeFactor where the target sizeFactor gets written to.

steps the current number of transform steps to be assumed (-1 for automatic determination)

8.73.3.8 virtual coeff WaveletTransform::saverage (int *subband*, area *channel*) const [virtual]

Return the average color for a particular area in a subband inside the image. The coefficients' signedness will be considered. This is a shortcut to the corresponding method in the Image class.

Parameters:

subband the subband

channel the area in that subband

Returns:

the average color

8.73.3.9 virtual coeff WaveletTransform::aaverage (int *subband*, area *channel*) const [virtual]

Return the average color for a particular area in a subband inside the image. The coefficients' signedness will be discarded. This is a shortcut to the corresponding method in the Image class.

Parameters:

subband the subband

channel the area in that subband

Returns:

the average color

8.73.3.10 virtual coeff WaveletTransform::sqvariance (int *subband*, area *channel*) const [virtual]

Return the square variance for a particular area in a subband inside the image. This is a shortcut to the corresponding method in the Image class.

Parameters:

subband the subband

channel the area in that subband

Returns:

the square variance

8.73.3.11 virtual coeff WaveletTransform::variance (int *subband*, area *channel*, bool *abs* = false) const [virtual]

Return the variance for a particular area in a subband inside the image. This is a shortcut to the corresponding method in the Image class.

Parameters:

subband the subband

channel the area in that subband

abs true if signs are discarded

Returns:

the square variance

8.73.3.12 virtual coeff WaveletTransform::sdeviation (int *subband*, area *channel*, bool *abs* = false) const [virtual]

Returns the standard deviation for a particular area in a subband inside the image. This is a shortcut to the corresponding method in the Image class.

Parameters:

subband the subband

channel the area in that subband

abs true if signs are discarded

Returns:

the standard deviation.

8.73.3.13 virtual coeff WaveletTransform::ratio (int *subband1*, int *subband2*) [virtual]

Return the ratio resulting from the zerotree relationship between two subbands (e.g. 1:4 from subband 1 to 2). This method depends on the geometry associated with the 'Pyramid' decomposition. Decomposition schemes using a different geometries, like e.g. the Packet transform may have to implement their own version of that method.

Parameters:

subband1 the first subband

subband2 the second subband

Returns:

the ratio, e.g. 4 for 1:4

8.73.3.14 Image* WaveletTransform::ll (int *steps* = -1) [inline]

Get a subpicture containing the LL subband.

Parameters:

steps assume number of transform steps instead of those computed from the previous transforms

Definition at line 184 of file WaveletTransform.hh.

References LL.

8.73.3.15 Image* WaveletTransform::hl (int *steps* = -1) [inline]

Get a subpicture containing the HL subband.

Parameters:

steps assume number of transform steps instead of those computed from the previous transforms

Definition at line 189 of file WaveletTransform.hh.

References HL.

8.73.3.16 Image* WaveletTransform::lh (int *steps* = -1) [inline]

Get a subpicture containing the LH subband.

Parameters:

steps assume number of transform steps instead of those computed from the previous transforms

Definition at line 194 of file WaveletTransform.hh.

References LH.

8.73.3.17 Image* WaveletTransform::hh (int *steps* = -1) [inline]

Get a subpicture containing the HH subband.

Parameters:

steps assume number of transform steps instead of those computed from the previous transforms

Definition at line 199 of file WaveletTransform.hh.

References HH.

8.73.3.18 Image* WaveletTransform::highMax (int *steps* = -1)

Get an image consisting of the given subband's highpass components maxima.

Parameters:

steps assume number of transform steps instead of those computed from

Returns:

a new image object containing the maxima as coefficients

8.73.3.19 Image& WaveletTransform::image (void) const [inline]

Get a reference to the image

Returns:

the reference to the image

Definition at line 210 of file WaveletTransform.hh.

8.73.3.20 virtual Image* WaveletTransform::subband (area *what*, int *steps* = -1) [virtual]

Get a subpicture containing one of the subbands.

Parameters:

what the subband, out of LL, HL, LH, HH

steps assume number of transform steps instead of those computed from the previous transforms

8.73.3.21 virtual int WaveletTransform::getSubband (int *ypos*, int *xpos*, int *steps* = -1) [virtual]

Get the subband of a given position in the image. This default method of calculation refers to the Pyramid transform and due to the lack of alternatives also to the Standard transform. Decomposition schemes using a different geometries, like e.g. the Packet transform may have to implement their own version of that method.

Parameters:

ypos the position's row

xpos the position's col

steps the number of steps we assume the image to have been transformed

Returns:

the position's subband

8.73.3.22 int WaveletTransform::getSubband (CoeffInformation & *c*, int *steps* = -1) [inline]

Get the subband of a given position in the image denoted by a {*CoeffInformation*} object. This default method of calculation refers to the Pyramid transform and due * to the lack of alternatives also to the Standard transform. Decomposition schemes using a different geometries, like e.g. the Packet transform may have to implement their own version of that method.

Parameters:

c a reference the {*CoeffInformation*} object.

steps the number of steps we assume the image to have been transformed

Returns:

the position's subband

Definition at line 246 of file WaveletTransform.hh.

References CoeffInformation::xpos(), and CoeffInformation::ypos().

8.73.3.23 virtual area WaveletTransform::getArea (int *ypos*, int *xpos*, int *steps* = -1) [virtual]

Get the area of a given position in the image. This default method of calculation refers to the Pyramid transform and due to the lack of alternatives also to the Standard transform. Decomposition schemes using a different geometries, like e.g. the Packet transform may have to implement their own version of that method.

Parameters:

ypos the position's row

xpos the position's col

steps the number of steps we assume the image to have been transformed

Returns:

the position's subband

8.73.3.24 area WaveletTransform::getArea (CoeffInformation & *c*, int *steps* = -1) [inline]

Get the area of a given position in the image denoted by a {*CoeffInformation*} object. This default method of calculation refers to the Pyramid transform and due $*$ to the lack of alternatives also to the Standard transform. Decomposition schemes using a different geometries, like e.g. the Packet transform may have to implement their own version of that method.

Parameters:

c a reference the {*CoeffInformation*} object.

steps the number of steps we assume the image to have been transformed

Returns:

the position's subband

Definition at line 276 of file WaveletTransform.hh.

References CoeffInformation::xpos(), and CoeffInformation::ypos().

8.73.3.25 virtual void WaveletTransform::import (Image & *img*, area *what*, int *steps* = -1) [virtual]

Import a subpicture containing one of the subbands.

Exceptions:

invalid_argument the imported image does not match the calculated subband size

Parameters:

img the image containing the subband

what the subband, out of LL, HL, LH, HH

steps assume number of transform steps instead of those computed from the previous transforms

8.73.3.26 virtual void WaveletTransform::fill (coeff *value*, area *what*, int *steps* = -1) [virtual]

Fill one of the subbands with one particular value.

Exceptions:

invalid_argument the imported image does not match the calculated subband size

Parameters:

value the value to be inserted

what the subband, out of LL, HL, LH, HH

steps assume number of transform steps instead of those computed from the previous transforms

8.73.3.27 int WaveletTransform::steps (void) const [inline]

Return the current number of decomposition steps

Returns:

the number of decomposition steps

Definition at line 306 of file WaveletTransform.hh.

8.73.3.28 void WaveletTransform::sanity (void) [private]

Resize m_buffer for transforms if image dimensions have changed

8.73.3.29 virtual void WaveletTransform::doanalysis (int *steps*) [protected, pure virtual]

Perform a Wavelet transform on the image.

Parameters:

steps the number of transform steps

Implemented in PyramidTransform, and StandardTransform.

8.73.3.30 virtual void WaveletTransform::dosynthesis (int *steps*, int *prevSteps* = 0) [protected, pure virtual]

Perform an inverse Wavelet transform on the image.

Parameters:

steps the number of inverse transform steps

prevSteps if greater than 0, the number of previous decomposition steps to assume. This is necessary to reconstruct images that are not square and have side lengths which are not powers of two.

Implemented in PyramidTransform, and StandardTransform.

8.73.4 Member Data Documentation

8.73.4.1 Wavelet* WaveletTransform::m_wavelet [protected]

A reference to the wavelet to be used.

Definition at line 318 of file WaveletTransform.hh.

8.73.4.2 Image* WaveletTransform::m_image [protected]

A reference to the image to be used.

Definition at line 320 of file WaveletTransform.hh.

8.73.4.3 Image* WaveletTransform::m_buffer [protected]

A second image used as temporary m_buffer.

Definition at line 322 of file WaveletTransform.hh.

8.73.4.4 FilterSet* WaveletTransform::m_filter [protected]

A reference to the filter set used.

Definition at line 324 of file WaveletTransform.hh.

8.73.4.5 int WaveletTransform::m_rows [protected]

The image's number of rows before resizing

Definition at line 326 of file WaveletTransform.hh.

8.73.4.6 int WaveletTransform::m_cols [protected]

The image's number of cols before resizing

Definition at line 328 of file WaveletTransform.hh.

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

- WaveletTransform.hh

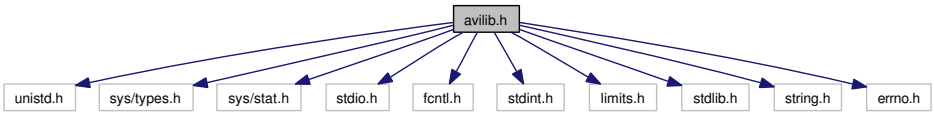
Chapter 9

File Documentation

9.1 avilib.h File Reference

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <fcntl.h>
#include <stdint.h>
#include <limits.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
```

Include dependency graph for avilib.h:



Classes

- struct video_index_entry
- struct audio_index_entry
- struct _avisuperindex_entry
- struct _avistdindex_entry
- struct _avistdindex_chunk

- struct _avisuperindex_chunk
- struct track_s
- struct alBITMAPINFOHEADER
- struct avi_t
- struct riff_struct
- struct chunk_struct
- struct common_struct
- struct wave_header
- struct AVIStreamHeader

Defines

- #define COMP_GCC
- #define SYS_UNIX
- #define SYS_LINUX
- #define AVI_MAX_TRACKS 8
- #define AVI_INDEX_OF_INDEXES 0x00
- #define AVI_INDEX_OF_CHUNKS 0x01
- #define AVI_INDEX_IS_DATA 0x80
- #define AVI_INDEX_2FIELD 0x01
- #define AVI_MODE_WRITE 0
- #define AVI_MODE_READ 1
- #define AVI_ERR_SIZELIM 1
- #define AVI_ERR_OPEN 2
- #define AVI_ERR_READ 3
- #define AVI_ERR_WRITE 4
- #define AVI_ERR_WRITE_INDEX 5
- #define AVI_ERR_CLOSE 6
- #define AVI_ERR_NOT_PERM 7
- #define AVI_ERR_NO_MEM 8
- #define AVI_ERR_NO_AVI 9
- #define AVI_ERR_NO_HDRL 10
- #define AVI_ERR_NO_MOVI 11
- #define AVI_ERR_NO_VIDS 12
- #define AVI_ERR_NO_IDX 13
- #define WAVE_FORMAT_UNKNOWN (0x0000)
- #define WAVE_FORMAT_PCM (0x0001)
- #define WAVE_FORMAT_ADPCM (0x0002)
- #define WAVE_FORMAT_IBM_CVSD (0x0005)
- #define WAVE_FORMAT_ALAW (0x0006)
- #define WAVE_FORMAT_MULAW (0x0007)
- #define WAVE_FORMAT_OKI_ADPCM (0x0010)
- #define WAVE_FORMAT_DVI_ADPCM (0x0011)
- #define WAVE_FORMAT_DIGISTD (0x0015)
- #define WAVE_FORMAT_DIGIFIX (0x0016)
- #define WAVE_FORMAT_YAMAHA_ADPCM (0x0020)

- #define WAVE_FORMAT_DSP_TRUESPEECH (0x0022)
- #define WAVE_FORMAT_GSM610 (0x0031)
- #define IBM_FORMAT_MULAW (0x0101)
- #define IBM_FORMAT_ALAW (0x0102)
- #define IBM_FORMAT_ADPCM (0x0103)

Typedefs

- typedef struct _avisuperindex_entry avisuperindex_entry
- typedef struct _avistdindex_entry avistdindex_entry
- typedef struct _avistdindex_chunk avistdindex_chunk
- typedef struct _avisuperindex_chunk avisuperindex_chunk
- typedef struct track_s track_t

Functions

- struct __attribute__ ((__packed__))
- avi_t * AVI_open_output_file (char *filename)
- void AVI_set_video (avi_t *AVI, int width, int height, double fps, char *compressor)
- void AVI_set_audio (avi_t *AVI, int channels, long rate, int bits, int format, long mp3rate)
- int AVI_write_frame (avi_t *AVI, char *data, long bytes, int keyframe)
- int AVI_dup_frame (avi_t *AVI)
- int AVI_write_audio (avi_t *AVI, char *data, long bytes)
- int AVI_append_audio (avi_t *AVI, char *data, long bytes)
- long AVI_bytes_remain (avi_t *AVI)
- int AVI_close (avi_t *AVI)
- long AVI_bytes_written (avi_t *AVI)
- avi_t * AVI_open_input_file (char *filename, int getIndex)
- avi_t * AVI_open_input_indexfile (char *filename, int getIndex, char *indexfile)
- avi_t * AVI_open_fd (int fd, int getIndex)
- avi_t * AVI_open_indexfd (int fd, int getIndex, char *indexfile)
- int avi_parse_input_file (avi_t *AVI, int getIndex)
- int avi_parse_index_from_file (avi_t *AVI, char *filename)
- long AVI_audio_mp3rate (avi_t *AVI)
- long AVI_audio_padrate (avi_t *AVI)
- long AVI_video_frames (avi_t *AVI)
- int AVI_video_width (avi_t *AVI)
- int AVI_video_height (avi_t *AVI)
- double AVI_frame_rate (avi_t *AVI)
- char * AVI_video_compressor (avi_t *AVI)
- int AVI_audio_channels (avi_t *AVI)
- int AVI_audio_bits (avi_t *AVI)
- int AVI_audio_format (avi_t *AVI)
- long AVI_audio_rate (avi_t *AVI)

- long AVI_audio_bytes (avi_t *AVI)
- long AVI_audio_chunks (avi_t *AVI)
- int AVI_can_read_audio (avi_t *AVI)
- long AVI_max_video_chunk (avi_t *AVI)
- long AVI_frame_size (avi_t *AVI, long frame)
- long AVI_audio_size (avi_t *AVI, long frame)
- int AVI_seek_start (avi_t *AVI)
- int AVI_set_video_position (avi_t *AVI, long frame)
- long AVI_get_video_position (avi_t *AVI, long frame)
- long AVI_read_frame (avi_t *AVI, char *vidbuf, int *keyframe)
- int AVI_set_audio_position (avi_t *AVI, long byte)
- int AVI_set_audio_bitrate (avi_t *AVI, long bitrate)
- long AVI_get_audio_position_index (avi_t *AVI)
- int AVI_set_audio_position_index (avi_t *AVI, long indexpos)
- long AVI_read_audio (avi_t *AVI, char *audbuf, long bytes)
- long AVI_read_audio_chunk (avi_t *AVI, char *audbuf)
- long AVI_audio_codech_offset (avi_t *AVI)
- long AVI_audio_codecf_offset (avi_t *AVI)
- long AVI_video_codech_offset (avi_t *AVI)
- long AVI_video_codecf_offset (avi_t *AVI)
- int AVI_read_data (avi_t *AVI, char *vidbuf, long max_vidbuf, char *audbuf, long max_audbuf, long *len)
- void AVI_print_error (char *str)
- char * AVI_strerror (void)
- char * AVI_syserror (void)
- int AVI_scan (char *name)
- int AVI_dump (char *name, int mode)
- char * AVI_codec2str (short cc)
- int AVI_file_check (char *import_file)
- void AVI_info (avi_t *avifile)
- uint64_t AVI_max_size (void)
- int avi_update_header (avi_t *AVI)
- int AVI_set_audio_track (avi_t *AVI, int track)
- int AVI_get_audio_track (avi_t *AVI)
- int AVI_audio_tracks (avi_t *AVI)
- void AVI_set_audio_vbr (avi_t *AVI, long is_vbr)
- long AVI_get_audio_vbr (avi_t *AVI)
- void AVI_set_comment_fd (avi_t *AVI, int fd)
- int AVI_get_comment_fd (avi_t *AVI)
- int AVI_read_wave_header (int fd, struct wave_header *wave)
- int AVI_write_wave_header (int fd, const struct wave_header *wave)
- size_t AVI_read_wave_pcm_data (int fd, void *buffer, size_t buflen)
- size_t AVI_write_wave_pcm_data (int fd, const void *buffer, size_t buflen)

Variables

- alWAVEFORMATEX
- alAVISTREAMINFO

9.1.1 Define Documentation

9.1.1.1 #define AVI_ERR_CLOSE 6

Definition at line 342 of file avilib.h.

9.1.1.2 #define AVI_ERR_NO_AVI 9

Definition at line 352 of file avilib.h.

9.1.1.3 #define AVI_ERR_NO_HDRL 10

Definition at line 354 of file avilib.h.

9.1.1.4 #define AVI_ERR_NO_IDX 13

Definition at line 362 of file avilib.h.

9.1.1.5 #define AVI_ERR_NO_MEM 8

Definition at line 350 of file avilib.h.

9.1.1.6 #define AVI_ERR_NO_MOVI 11

Definition at line 357 of file avilib.h.

9.1.1.7 #define AVI_ERR_NO_VIDS 12

Definition at line 360 of file avilib.h.

9.1.1.8 #define AVI_ERR_NOT_PERM 7

Definition at line 346 of file avilib.h.

9.1.1.9 #define AVI_ERR_OPEN 2

Definition at line 330 of file avilib.h.

9.1.1.10 #define AVI_ERR_READ 3

Definition at line 333 of file avilib.h.

9.1.1.11 #define AVI_ERR_SIZELIM 1

Definition at line 325 of file avilib.h.

9.1.1.12 #define AVI_ERR_WRITE 4

Definition at line 335 of file avilib.h.

9.1.1.13 #define AVI_ERR_WRITE_INDEX 5

Definition at line 338 of file avilib.h.

9.1.1.14 #define AVI_INDEX_2FIELD 0x01

Definition at line 145 of file avilib.h.

9.1.1.15 #define AVI_INDEX_IS_DATA 0x80

Definition at line 141 of file avilib.h.

9.1.1.16 #define AVI_INDEX_OF_CHUNKS 0x01

Definition at line 139 of file avilib.h.

9.1.1.17 #define AVI_INDEX_OF_INDEXES 0x00

Definition at line 137 of file avilib.h.

9.1.1.18 #define AVI_MAX_TRACKS 8

Definition at line 117 of file avilib.h.

9.1.1.19 #define AVI_MODE_READ 1

Definition at line 321 of file avilib.h.

9.1.1.20 #define AVI_MODE_WRITE 0

Definition at line 320 of file avilib.h.

9.1.1.21 #define COMP_GCC

Definition at line 47 of file avilib.h.

9.1.1.22 #define IBM_FORMAT_ADPCM (0x0103)

Definition at line 384 of file avilib.h.

9.1.1.23 #define IBM_FORMAT_ALAW (0x0102)

Definition at line 383 of file avilib.h.

9.1.1.24 #define IBM_FORMAT_MULAW (0x0101)

Definition at line 382 of file avilib.h.

9.1.1.25 #define SYS_LINUX

Definition at line 52 of file avilib.h.

9.1.1.26 #define SYS_UNIX

Definition at line 48 of file avilib.h.

9.1.1.27 #define WAVE_FORMAT_ADPCM (0x0002)

Definition at line 371 of file avilib.h.

9.1.1.28 #define WAVE_FORMAT_ALAW (0x0006)

Definition at line 373 of file avilib.h.

9.1.1.29 #define WAVE_FORMAT_DIGIFIX (0x0016)

Definition at line 378 of file avilib.h.

9.1.1.30 #define WAVE_FORMAT_DIGISTD (0x0015)

Definition at line 377 of file avilib.h.

9.1.1.31 #define WAVE_FORMAT_DSP_TRUESPEECH (0x0022)

Definition at line 380 of file avilib.h.

9.1.1.32 #define WAVE_FORMAT_DVI_ADPCM (0x0011)

Definition at line 376 of file avilib.h.

9.1.1.33 #define WAVE_FORMAT_GSM610 (0x0031)

Definition at line 381 of file avilib.h.

9.1.1.34 #define WAVE_FORMAT_IBM_CVSD (0x0005)

Definition at line 372 of file avilib.h.

9.1.1.35 #define WAVE_FORMAT_MULAW (0x0007)

Definition at line 374 of file avilib.h.

9.1.1.36 #define WAVE_FORMAT_OKI_ADPCM (0x0010)

Definition at line 375 of file avilib.h.

9.1.1.37 #define WAVE_FORMAT_PCM (0x0001)

Definition at line 370 of file avilib.h.

9.1.1.38 #define WAVE_FORMAT_UNKNOWN (0x0000)

Definition at line 369 of file avilib.h.

9.1.1.39 #define WAVE_FORMAT_YAMAHA_ADPCM (0x0020)

Definition at line 379 of file avilib.h.

9.1.2 Typedef Documentation

9.1.2.1 typedef struct _avistdindex_chunk avistdindex_chunk

9.1.2.2 typedef struct _avistdindex_entry avistdindex_entry

9.1.2.3 typedef struct _avisuperindex_chunk avisuperindex_chunk

9.1.2.4 typedef struct _avisuperindex_entry avisuperindex_entry

9.1.2.5 typedef struct track_s track_t

9.1.3 Function Documentation

9.1.3.1 struct __attribute__ ((__packed__)) [read]

Definition at line 235 of file avilib.h.

9.1.3.2 int AVI_append_audio (avi_t * *AVI*, char * *data*, long *bytes*)

9.1.3.3 int AVI_audio_bits (avi_t * *AVI*)

9.1.3.4 long AVI_audio_bytes (avi_t * *AVI*)

9.1.3.5 int AVI_audio_channels (avi_t * *AVI*)

9.1.3.6 long AVI_audio_chunks (avi_t * *AVI*)

9.1.3.7 long AVI_audio_codecf_offset (avi_t * *AVI*)

9.1.3.8 long AVI_audio_codech_offset (avi_t * *AVI*)

9.1.3.9 int AVI_audio_format (avi_t * *AVI*)

9.1.3.10 long AVI_audio_mp3rate (avi_t * *AVI*)

9.1.3.11 long AVI_audio_padrate (avi_t * *AVI*)

9.1.3.12 long AVI_audio_rate (avi_t * *AVI*)

9.1.3.13 long AVI_audio_size (avi_t * *AVI*, long *frame*)

9.1.3.14 int AVI_audio_tracks (avi_t * *AVI*)

9.1.3.15 long AVI_bytes_remain (avi_t * *AVI*)

9.1.3.16 long AVI_bytes_written (avi_t * *AVI*)

9.1.3.17 int AVI_can_read_audio (avi_t * *AVI*)

9.1.3.18 int AVI_close (avi_t * *AVI*)

9.1.3.19 char* AVI_codec2str (short *cc*)

9.1.3.20 int AVI_dump (char * *name*, int *mode*)

9.1.3.21 int AVI_dup_frame (avi_t * *AVI*)

9.1.3.22 int AVI_file_check (char * *import_file*)

9.1.3.23 double AVI_frame_rate (avi_t * *AVI*)

9.1.3.24 long AVI_frame_size (avi_t * *AVI*, long *frame*)

9.1.3.25 long AVI_get_audio_position_index (avi_t * *AVI*)

9.1.3.26 int AVI_get_audio_track (avi_t * *AVI*)

9.1.3.27 long AVI_get_audio_vbr (avi_t * *AVI*)

9.1.3.28 int AVI_get_comment_fd (avi_t * *AVI*)

9.1.3.29 long AVI_get_video_position (avi_t * *AVI*, long *frame*)

9.1.3.30 void AVI_info (avi_t * *avifile*)

9.1.3.31 uint64_t AVI_max_size (void)

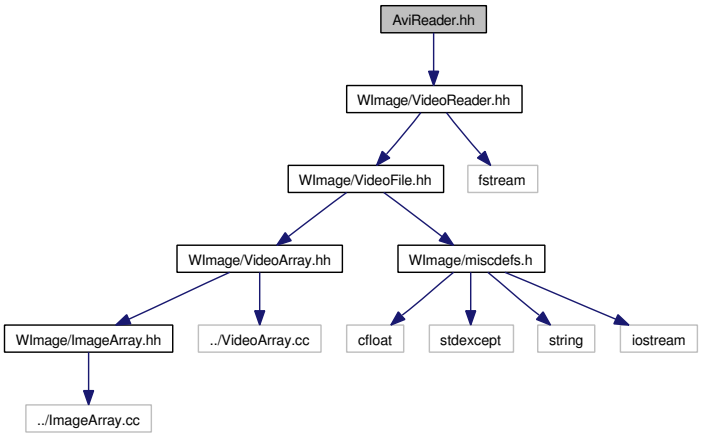
9.1.4.2 alWAVEFORMATEX

Definition at line 244 of file avilib.h.

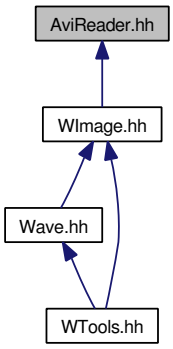
9.2 AviReader.hh File Reference

`#include "WImage/VideoReader.hh"`

Include dependency graph for AviReader.hh:



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



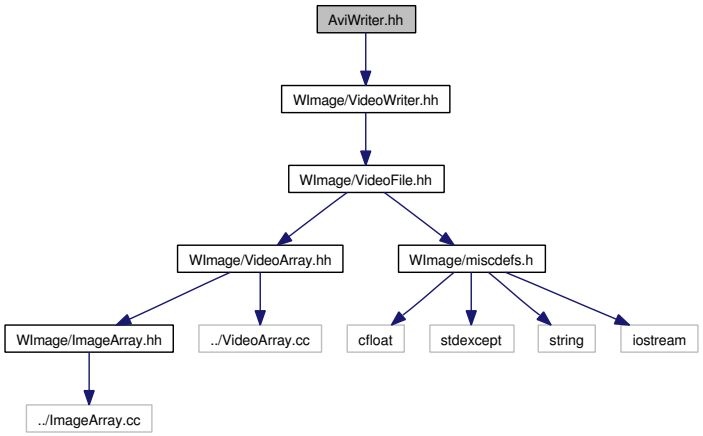
Classes

- class AviReader

9.3 AviWriter.hh File Reference

```
#include "WImage/VideoWriter.hh"
```

Include dependency graph for AviWriter.hh:



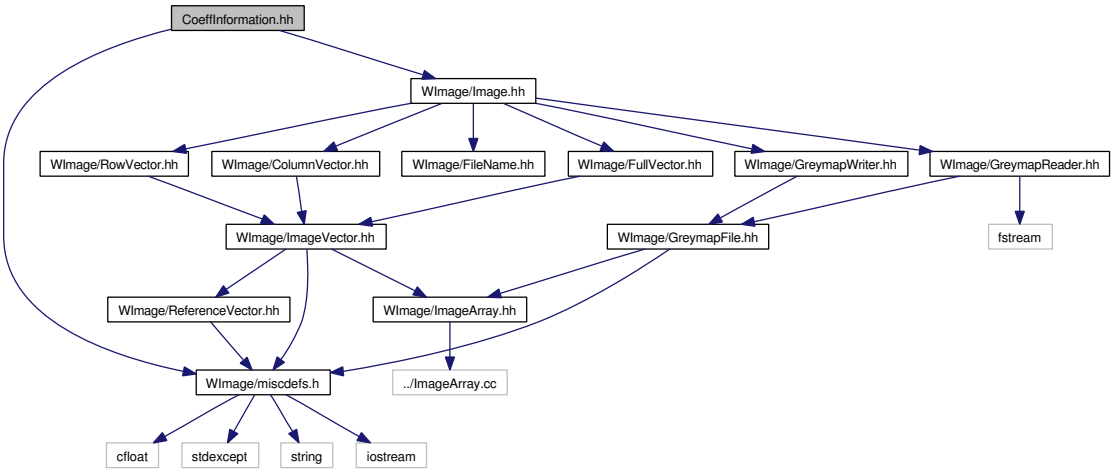
Classes

- class AviWriter

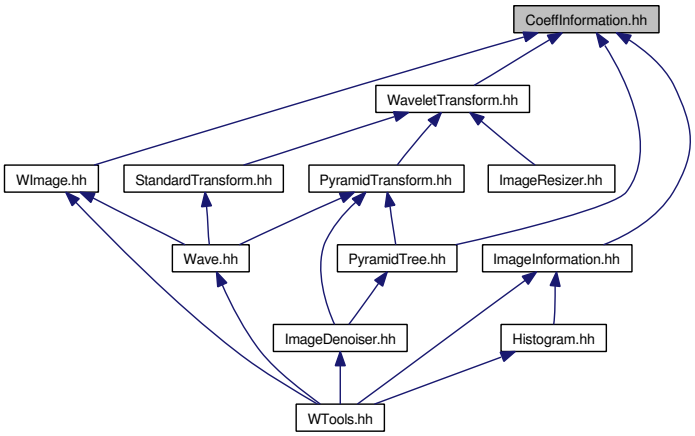
9.4 CoeffInformation.hh File Reference

```
#include "WImage/miscdefs.h"
#include "WImage/Image.hh"
```

Include dependency graph for CoeffInformation.hh:



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



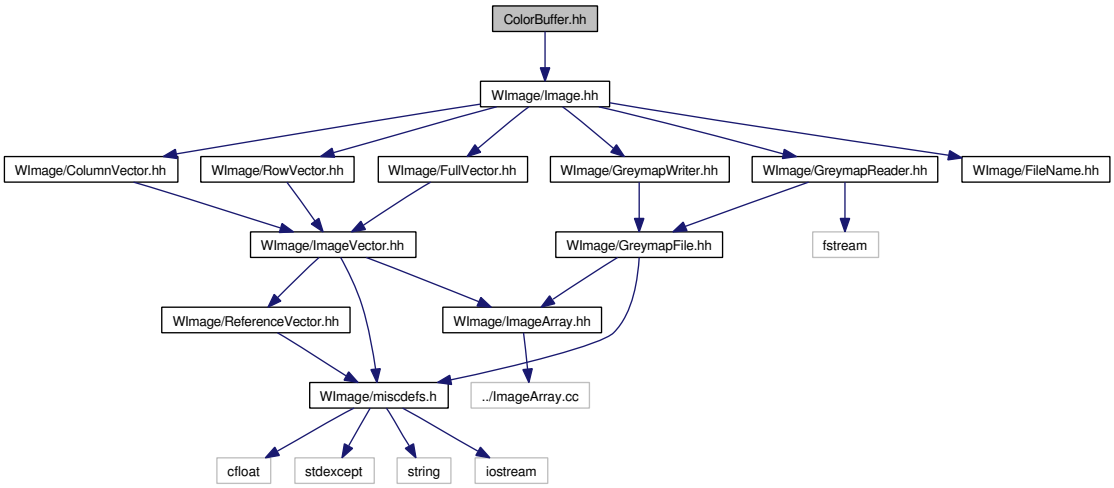
Classes

- class CoeffInformation

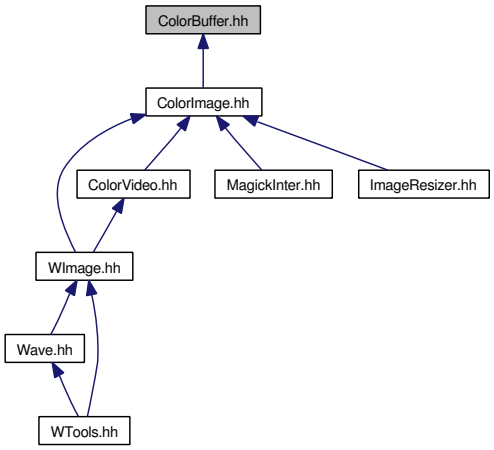
9.5 ColorBuffer.hh File Reference

```
#include "WImage/Image.hh"
```

Include dependency graph for ColorBuffer.hh:



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



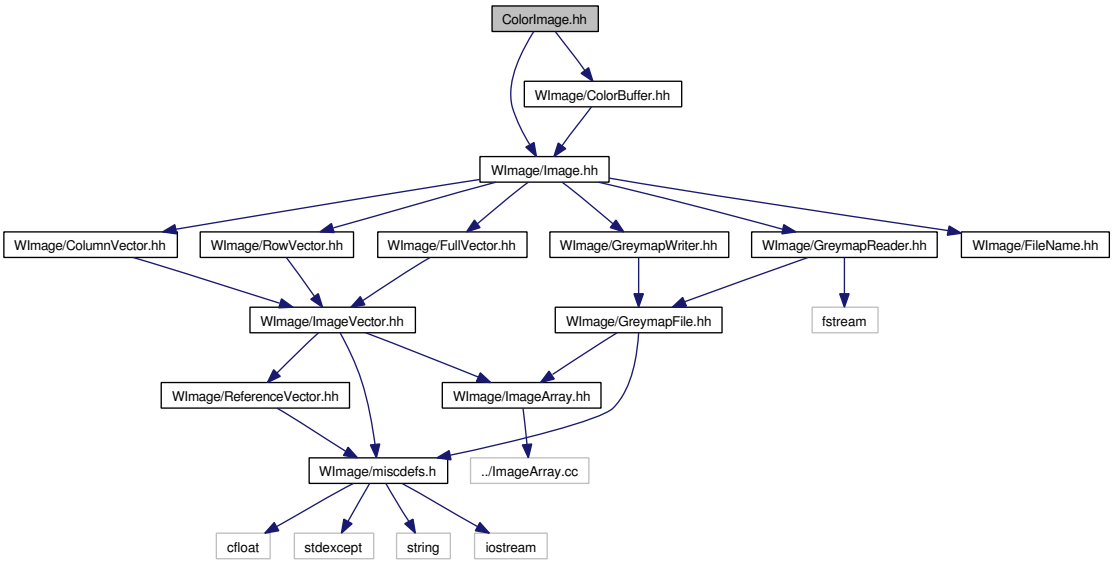
Classes

- class ColorBuffer

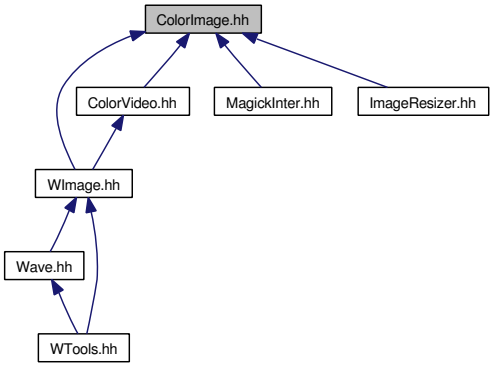
9.6 ColorImage.hh File Reference

```
#include "WImage/Image.hh"
#include "WImage/ColorBuffer.hh"
```

Include dependency graph for ColorImage.hh:



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



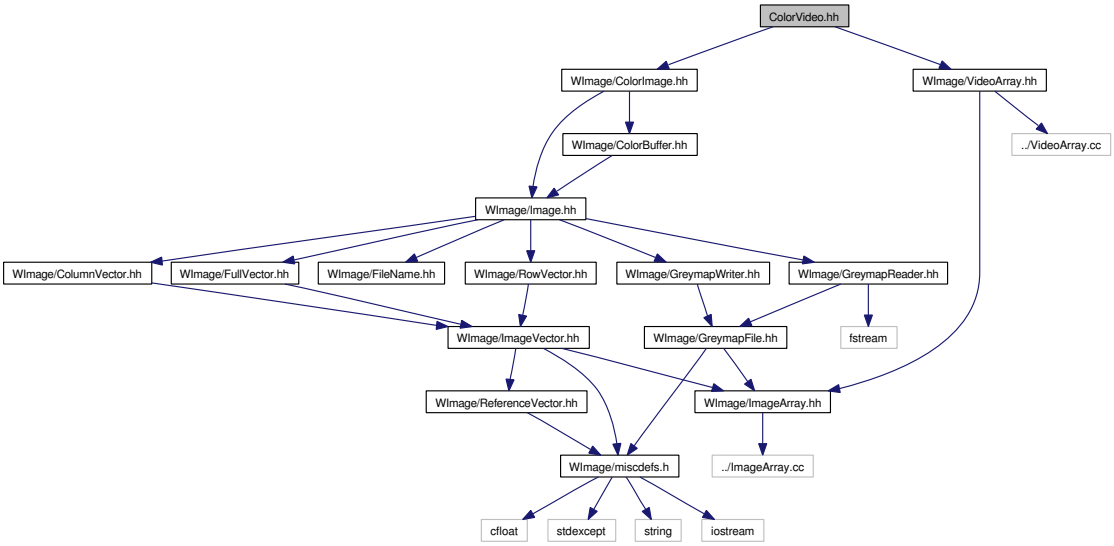
Classes

- class ColorImage

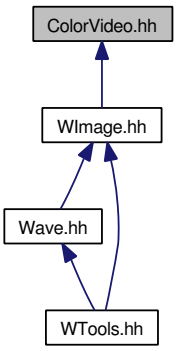
9.7 ColorVideo.hh File Reference

```
#include "WImage/ColorImage.hh"
#include "WImage/VideoArray.hh"
```

Include dependency graph for ColorVideo.hh:



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



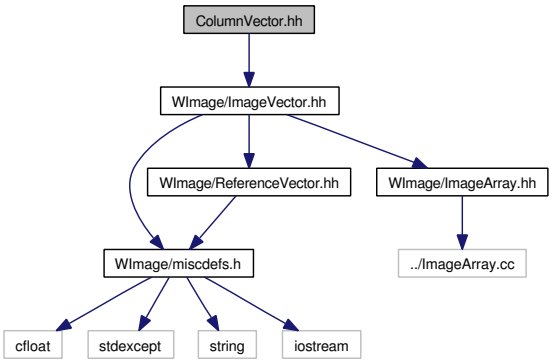
Classes

- class ColorVideo

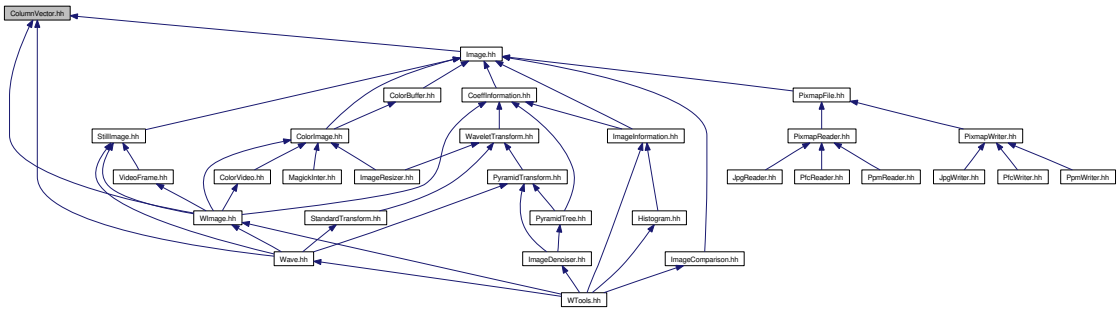
9.8 ColumnVector.hh File Reference

`#include "WImage/ImageVector.hh"`

Include dependency graph for ColumnVector.hh:



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



Classes

- class ColumnVector

9.9 debug.h File Reference

Defines

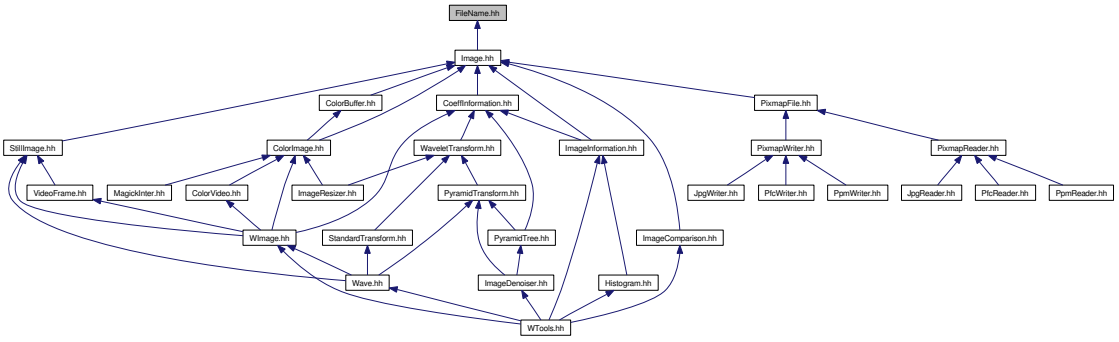
- #define STRR(X) #X
- #define STR(X) STRR(X)
- #define DPRINTF(X) (debug_printf("DEBUG %s [%s]: ", __FILE__, STR(__LINE__)), debug_printf X)

Functions

- int debug_printf (const char *tmpl,...)

9.10 FileName.hh File Reference

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



Classes

- class FileName

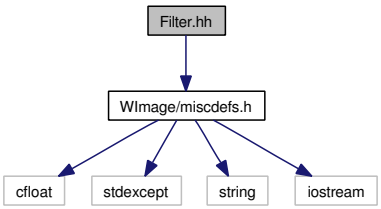
Enumerations

- enum filetype {
  fn_pgm, fn_raw, fn_pfi, fn_ppm,
  fn_jpg, fn_vid, fn_avi, fn_unknown }

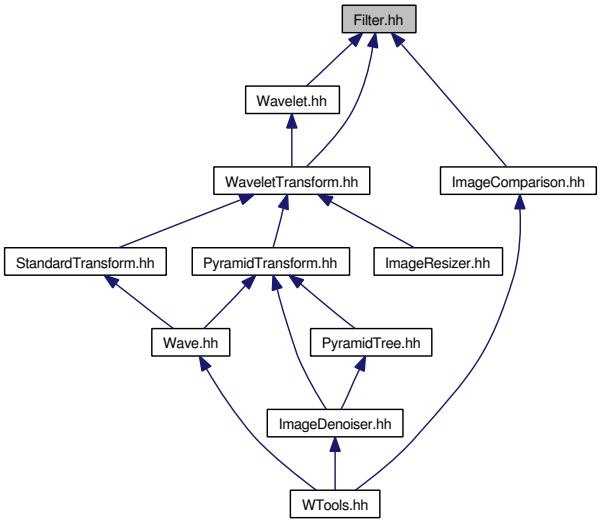
9.11 Filter.hh File Reference

```
#include "WImage/miscdefs.h"
```

Include dependency graph for Filter.hh:



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



Classes

- class Filter
- class FilterSet

Defines

- #define NULL 0

Variables

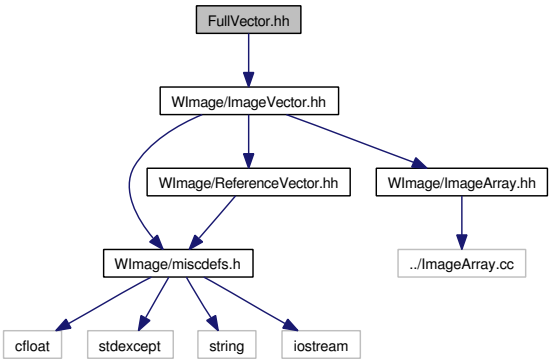
- FilterSet Haar

- FilterSet Daub4
- FilterSet Daub6
- FilterSet Daub8
- FilterSet Antonini
- FilterSet Brislawn
- FilterSet Villa1
- FilterSet Villa2
- FilterSet Villa3
- FilterSet Villa4
- FilterSet Villa5
- FilterSet Villa6
- FilterSet Odegard

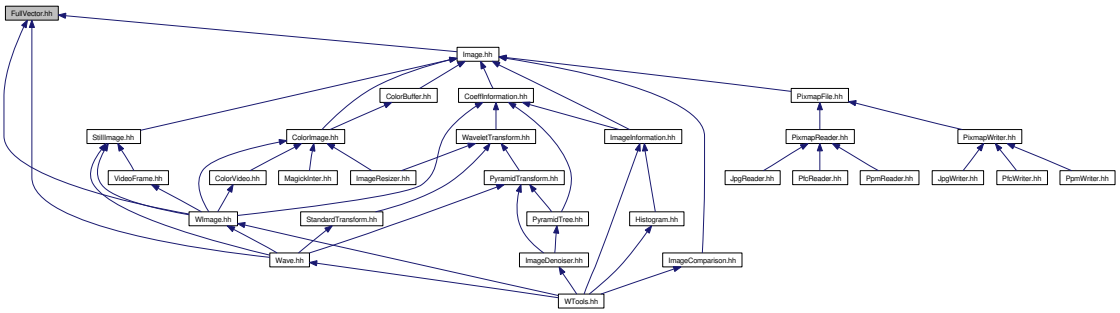
9.12 FullVector.hh File Reference

```
#include "WImage/ImageVector.hh"
```

Include dependency graph for FullVector.hh:



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



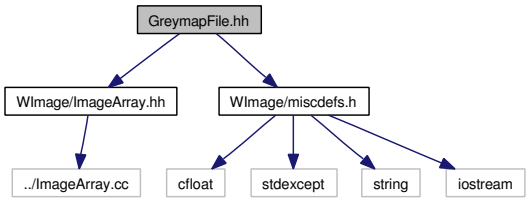
Classes

- class FullVector

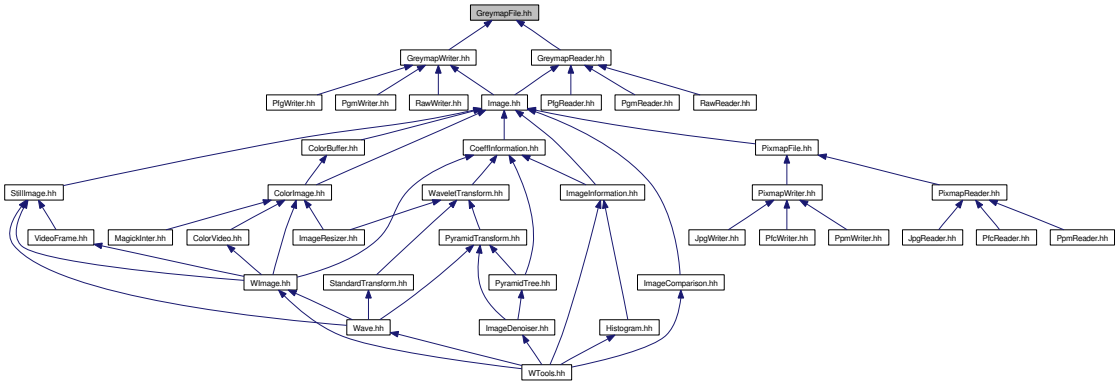
9.13 GreymapFile.hh File Reference

```
#include "WImage/ImageArray.hh"
#include "WImage/miscdefs.h"
```

Include dependency graph for GreymapFile.hh:



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



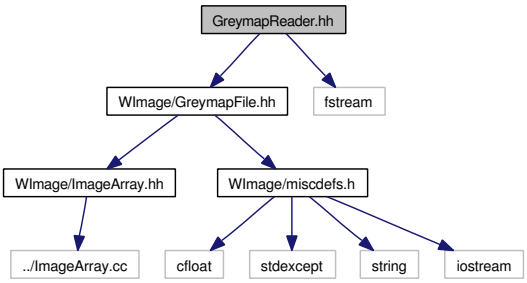
Classes

- class GreymapFile

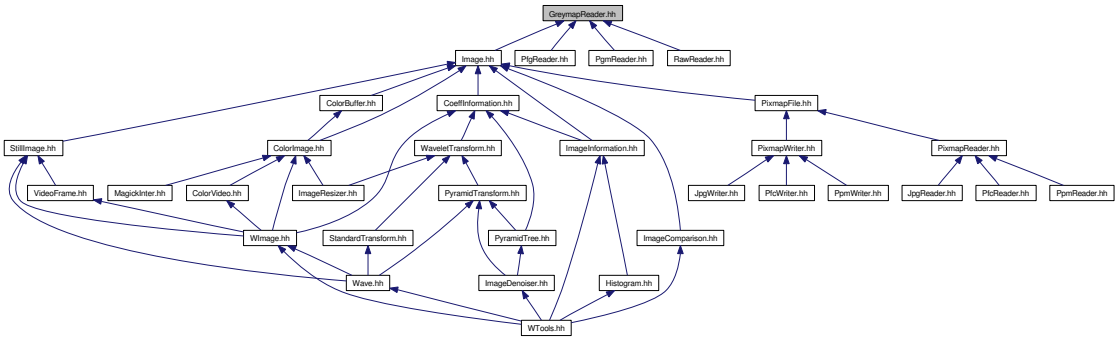
9.14 GreymapReader.hh File Reference

```
#include "WImage/GreymapFile.hh"
#include <fstream>
```

Include dependency graph for GreymapReader.hh:



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



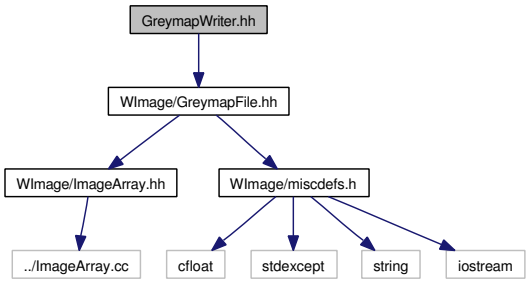
Classes

- class GreymapReader

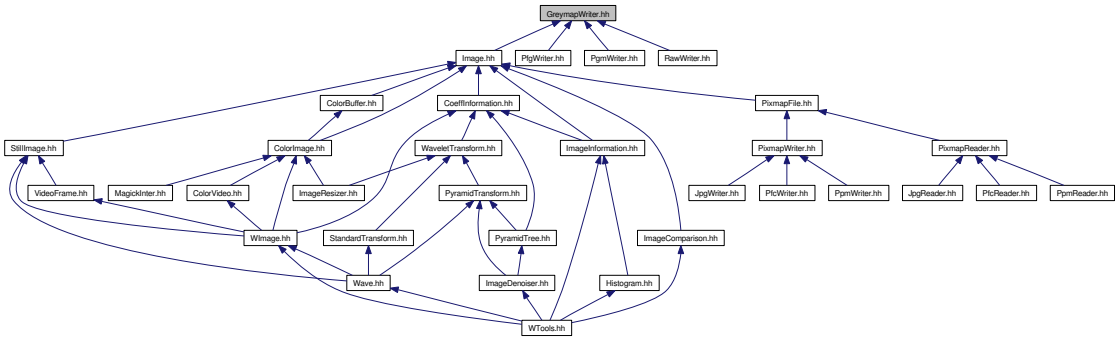
9.15 GreymapWriter.hh File Reference

`#include "WImage/GreymapFile.hh"`

Include dependency graph for GreymapWriter.hh:



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



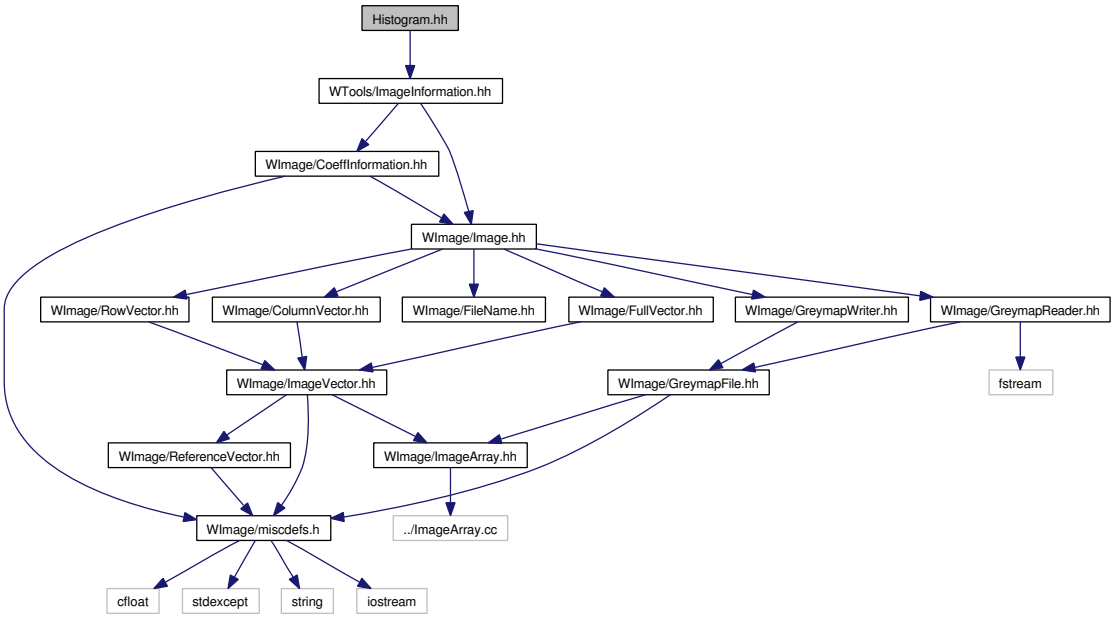
Classes

- class GreymapWriter

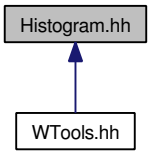
9.16 Histogram.hh File Reference

```
#include "WTools/ImageInformation.hh"
```

Include dependency graph for Histogram.hh:



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



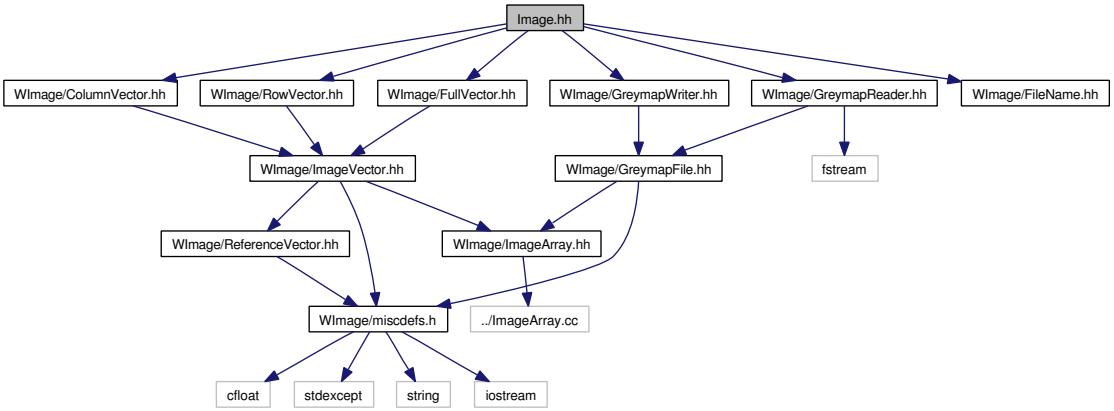
Classes

- class Histogram
- struct Histogram::hist

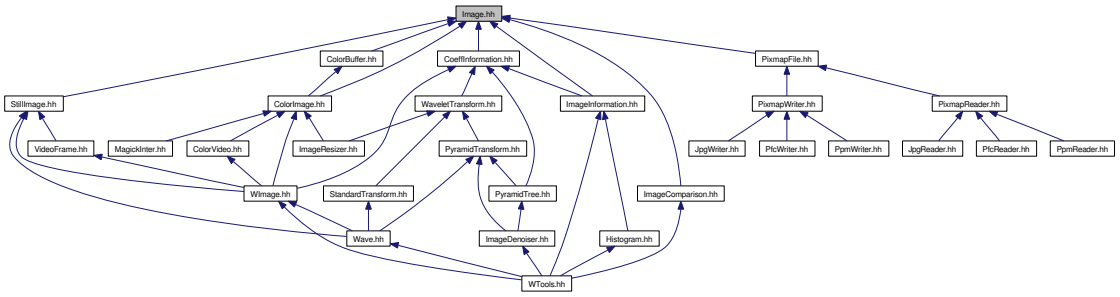
9.17 Image.hh File Reference

```
#include "WImage/ColumnVector.hh"
#include "WImage/RowVector.hh"
#include "WImage/FullVector.hh"
#include "WImage/GreymapReader.hh"
#include "WImage/GreymapWriter.hh"
#include "WImage/FileName.hh"
```

Include dependency graph for Image.hh:



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



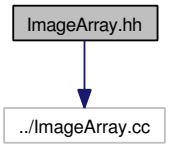
Classes

- class Image

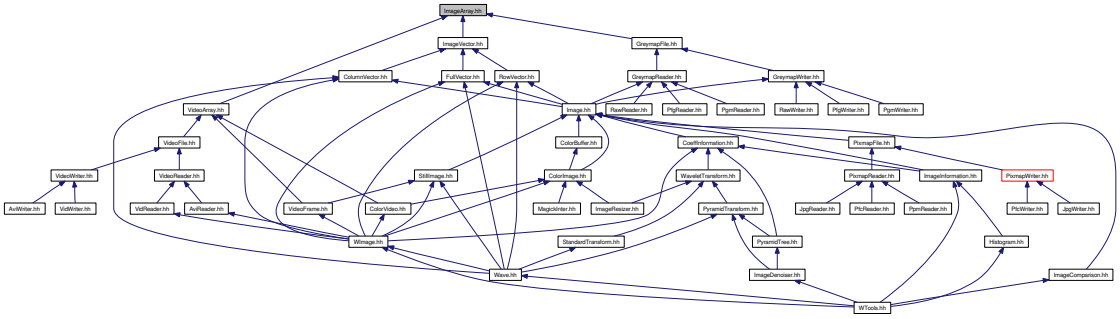
9.18 ImageArray.hh File Reference

```
#include "../ImageArray.cc"
```

Include dependency graph for ImageArray.hh:



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



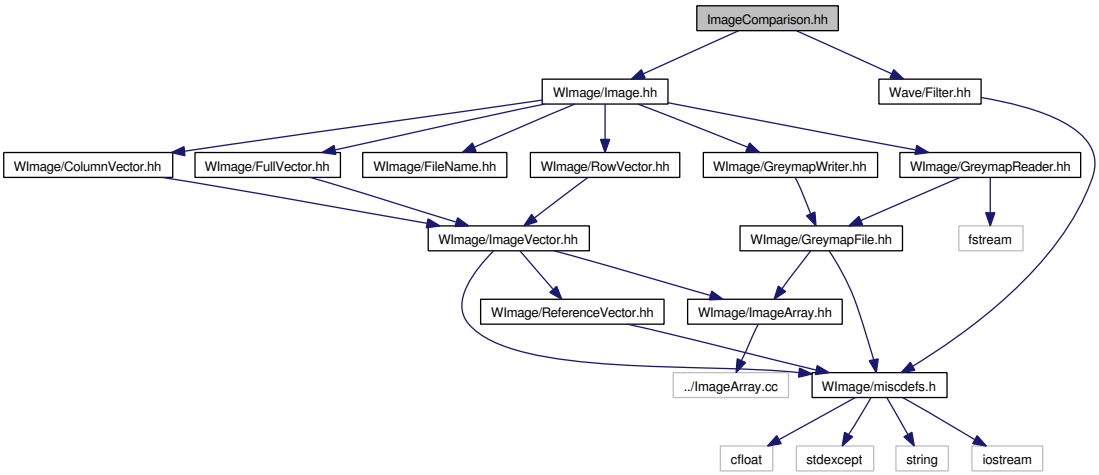
Classes

- class ImageArray< T >

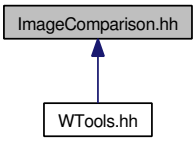
9.19 ImageComparison.hh File Reference

```
#include "WImage/Image.hh"
#include "Wave/Filter.hh"
```

Include dependency graph for ImageComparison.hh:



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



Classes

- struct lq
- struct logvals
- class ImageComparison

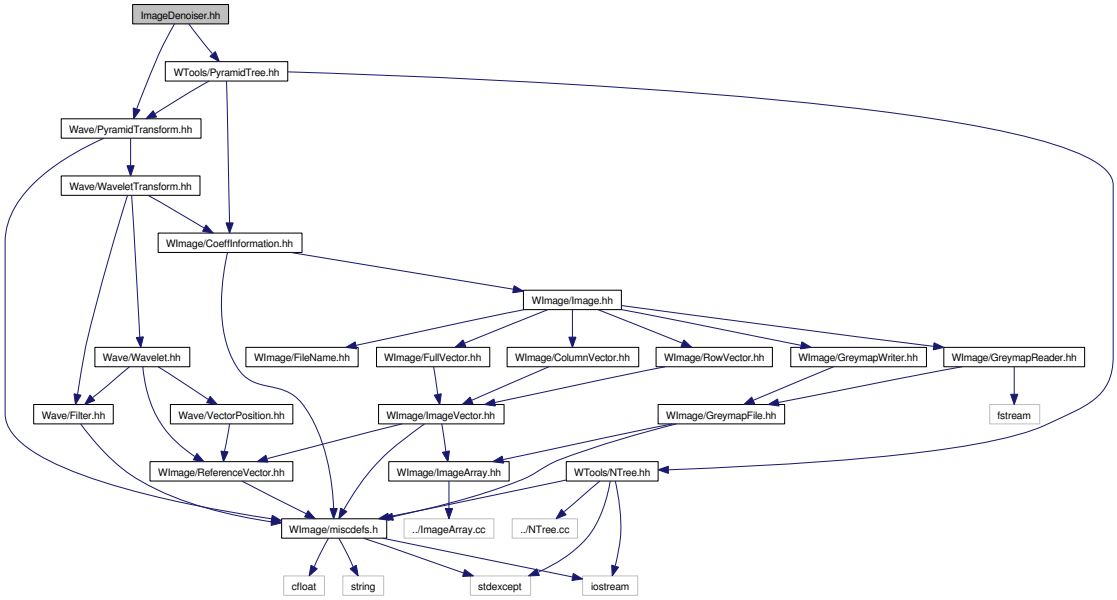
Enumerations

- enum imgtype { DRAWN = 0, SCANNED }

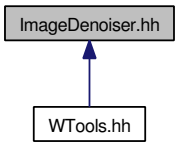
9.20 ImageDenoiser.hh File Reference

```
#include "Wave/PyramidTransform.hh"
#include "WTools/PyramidTree.hh"
```

Include dependency graph for ImageDenoiser.hh:



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



Classes

- class ImageDenoiser

Defines

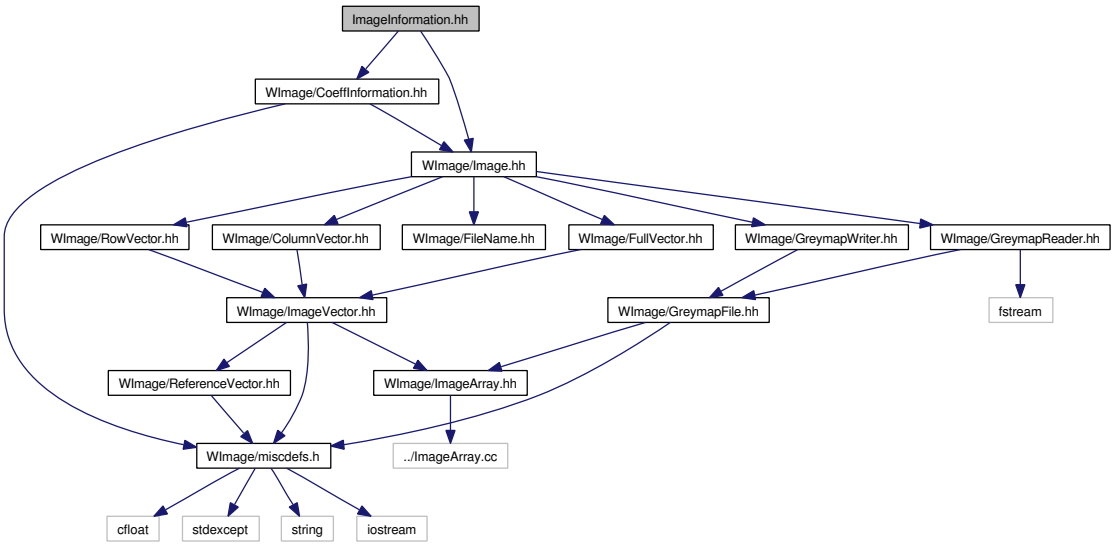
- #define DENOISE_HL 0x01
- #define DENOISE_LH 0x02
- #define DENOISE_HH 0x04
- #define SIGNIFICANT_COEFF 0

- #define SIGNIFICANT_REGION 1
- #define SIGNIFICANT_CHANNEL 2
- #define REPLACE_SIMPLE 0
- #define REPLACE_CHANNEL 1

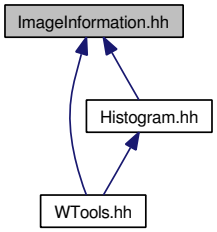
9.21 ImageInformation.hh File Reference

```
#include "WImage/CoeffInformation.hh"
#include "WImage/Image.hh"
```

Include dependency graph for ImageInformation.hh:



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



Classes

- class ImageInformation

Defines

- #define PII_YPOS(info, pos) (((info) → at (pos)).ypos ())
- #define PII_XPOS(info, pos) (((info) → at (pos)).xpos ())
- #define PII_XYPOS(info, pos) (((info) → at (pos)).xypos ())

- #define II_YPOS(info, pos) (((info).at (pos)).ypos ())
- #define II_XPOS(info, pos) (((info).at (pos)).xpos ())
- #define II_XYPOS(info, pos) (((info).at (pos)).xypos ())

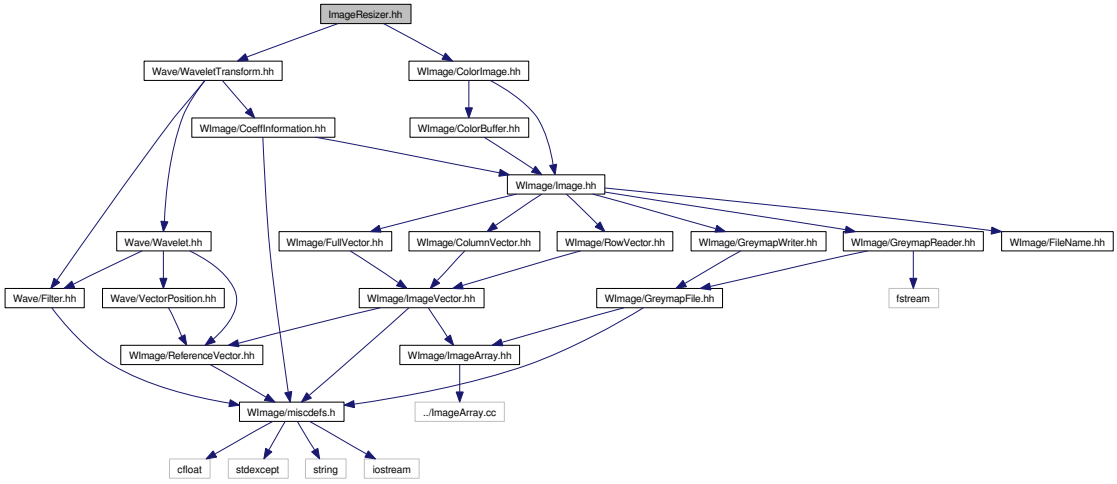
Typedefs

- typedef bool(∗ cipredicate)(const CoeffInformation ∗c1, const CoeffInformation ∗c2)

9.22 ImageResizer.hh File Reference

```
#include "WImage/ColorImage.hh"
#include "Wave/WaveletTransform.hh"
```

Include dependency graph for ImageResizer.hh:



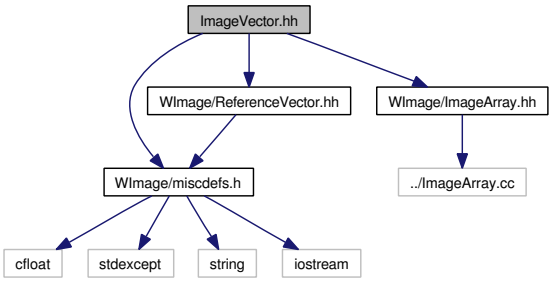
Classes

- class ImageResizer

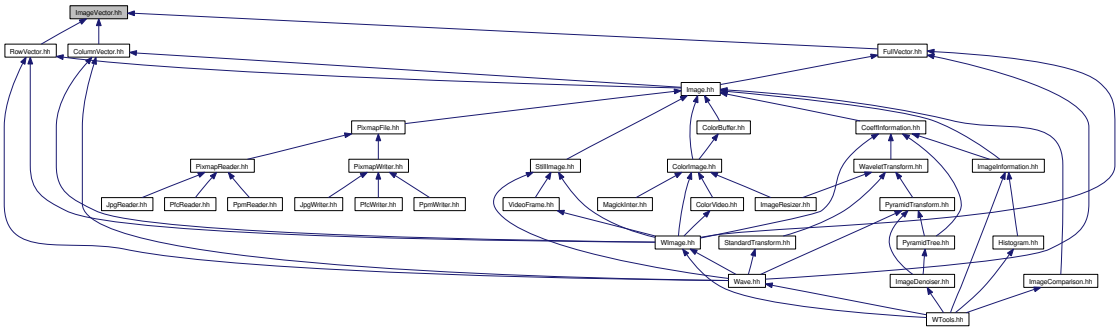
9.23 ImageVector.hh File Reference

```
#include "WImage/miscdefs.h"
#include "WImage/ImageArray.hh"
#include "WImage/ReferenceVector.hh"
```

Include dependency graph for ImageVector.hh:



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



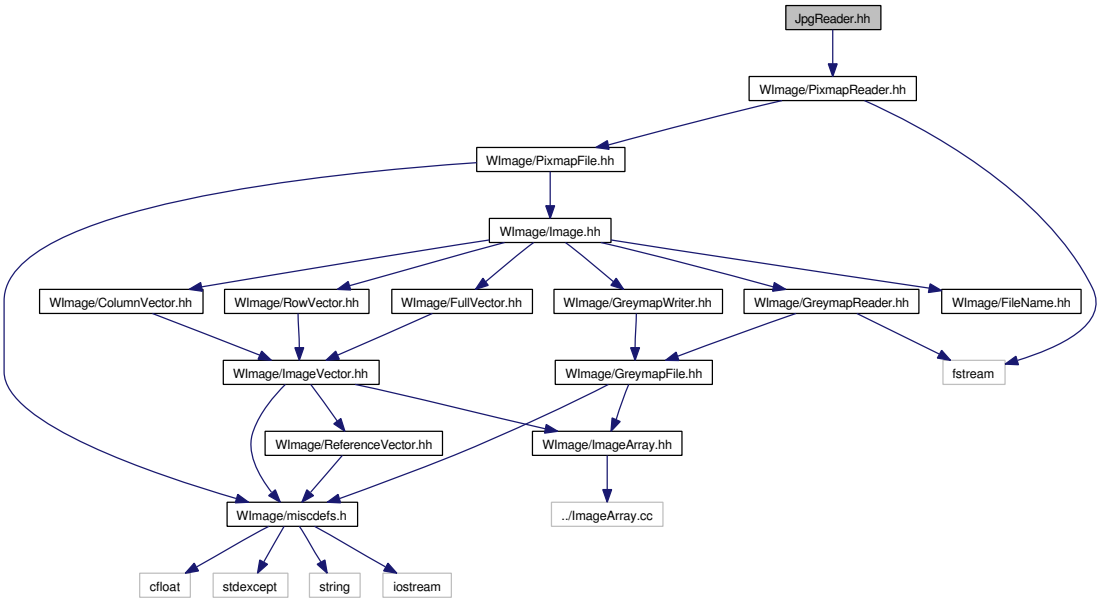
Classes

- class ImageVector

9.24 JpgReader.hh File Reference

```
#include "WImage/PixmapReader.hh"
```

Include dependency graph for JpgReader.hh:



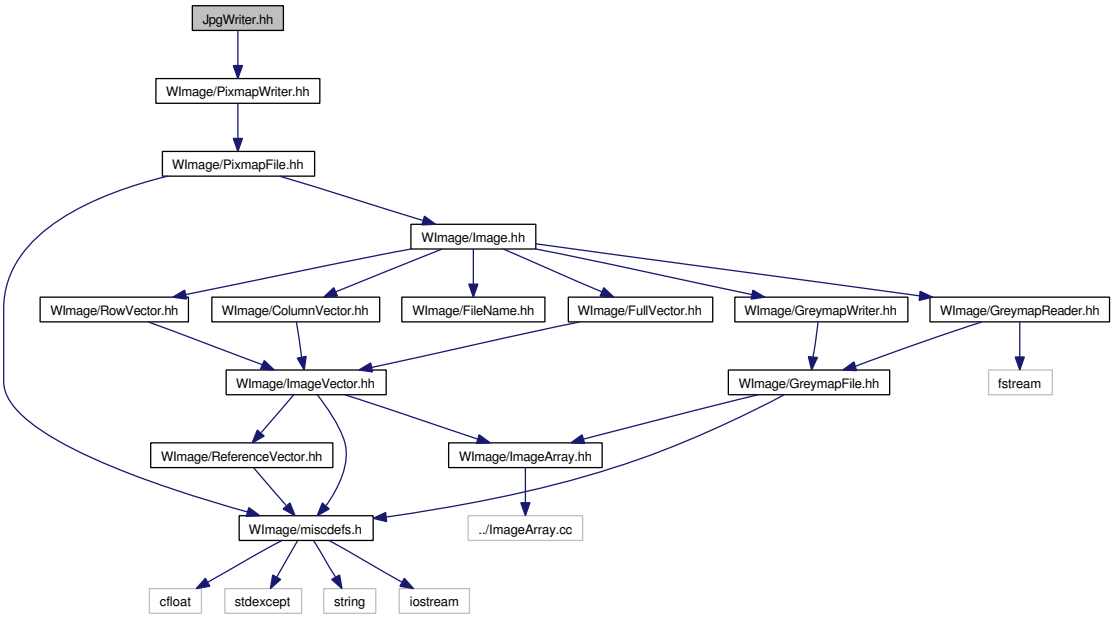
Classes

- class JpgReader

9.25 JpgWriter.hh File Reference

```
#include "WImage/PixmapWriter.hh"
```

Include dependency graph for JpgWriter.hh:



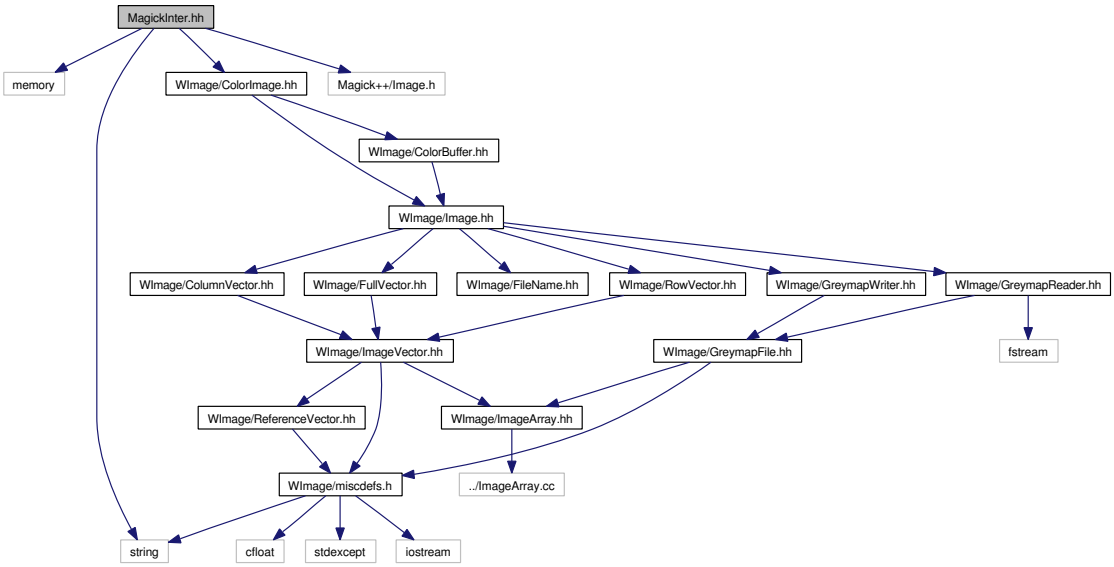
Classes

- class JpgWriter

9.26 MagickInter.hh File Reference

```
#include <memory>
#include <string>
#include <WImage/ColorImage.hh>
#include <Magick++/Image.h>
```

Include dependency graph for MagickInter.hh:



Namespaces

- namespace MagickInter

Functions

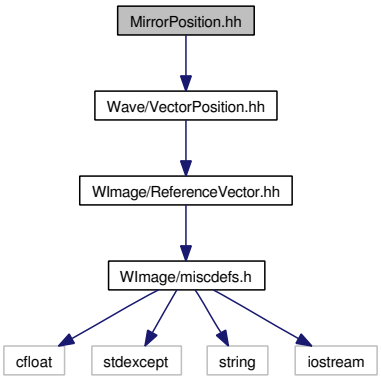
- Magick::Image MagickInter::magickImageFromColorImageWithTransparency (ColorImage &img, bool withTransparency=false, coeff *transparentColors=NULL, int colorBytes=1)
- Magick::Image MagickInter::magickImageFromColorImage (ColorImage &img)
- std::auto_ptr< ColorImage > MagickInter::colorImageFromMagickImage (Magick::Image &img)
- std::auto_ptr< ColorImage > MagickInter::obtainColorImage (const std::string &inFile)
- void MagickInter::writeColorImage (ColorImage &img, const std::string &outFile, int quality=100)

- void MagickInter::writeColorImageWithTransparency (ColorImage &img, const std::string &outFile, int quality=100, bool withTransparency=false, coeff *transparentColors=NULL, int colorBytes=1)
- void MagickInter::scaleAndWriteColorImage (ColorImage &img, int rows, int cols, const std::string &outFile, int quality=100)

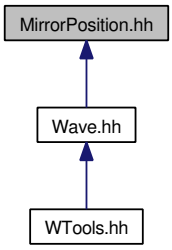
9.27 MirrorPosition.hh File Reference

```
#include "Wave/VectorPosition.hh"
```

Include dependency graph for MirrorPosition.hh:



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



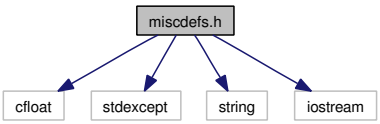
Classes

- class MirrorPosition

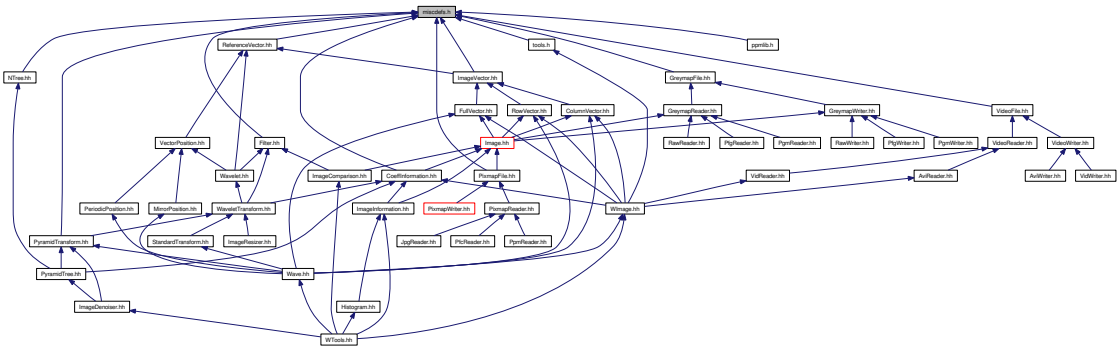
9.28 miscdefs.h File Reference

```
#include <cfloat>
#include <stdexcept>
#include <string>
#include <iostream>
```

Include dependency graph for miscdefs.h:



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



Defines

- #define NEW(a) new a
- #define DELETE(a) delete (a), (a) = NULL
- #define DELETEAR(a) delete [ ] (a), (a) = NULL
- #define DELETENOTNULL(a) if (a) { DELETE (a); } else {}
- #define DELETENOTNULLAR(a) if (a) { DELETEAR (a); } else {}
- #define COEFF_EPSILON ((coeff)DBL_EPSILON)
- #define COEFF_MAX ((coeff)DBL_MAX)
- #define COEFF_MIN ((coeff)DBL_MIN)
- #define MIN(x, y) ((x) < (y)? (x): (y))
- #define MAX(x, y) ((x) > (y)? (x): (y))
- #define DIV2(x) ((unsigned)(x)>>1u)
- #define MUL2(x) ((unsigned)(x)<<1u)
- #define TWOPOW(n) (1u<<(unsigned)(n))
- #define SQUARE(n) ((n)*(n))
- #define FMT0 0

Typedefs

- typedef unsigned char pixel
- typedef double coeff

Enumerations

- enum area {
  LL = 0, HL, LH, HH,
  areaINVALID }
- enum clrmodel { cm_rgb, cm_yuv, cm_grey, cm_unknown }
- enum yuv { yuv_y = 0, yuv_u, yuv_v, yuv_unknown }
- enum rgb { rgb_r = 0, rgb_g, rgb_b, rgb_unknown }

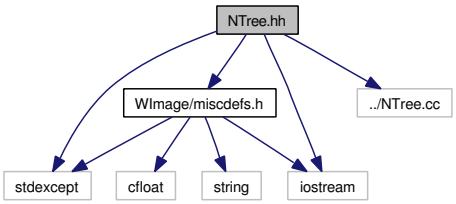
Variables

- const area areas [areaINVALID]

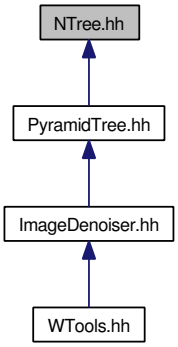
9.29 NTree.hh File Reference

```
#include "WImage/miscdefs.h"
#include <stdexcept>
#include <iostream>
#include "../NTree.cc"
```

Include dependency graph for NTree.hh:



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



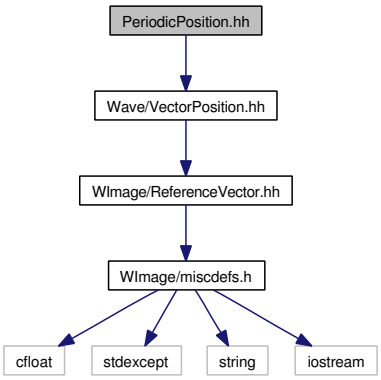
Classes

- class NTree< Type >

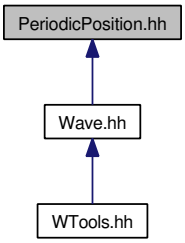
9.30 PeriodicPosition.hh File Reference

```
#include "Wave/VectorPosition.hh"
```

Include dependency graph for PeriodicPosition.hh:



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



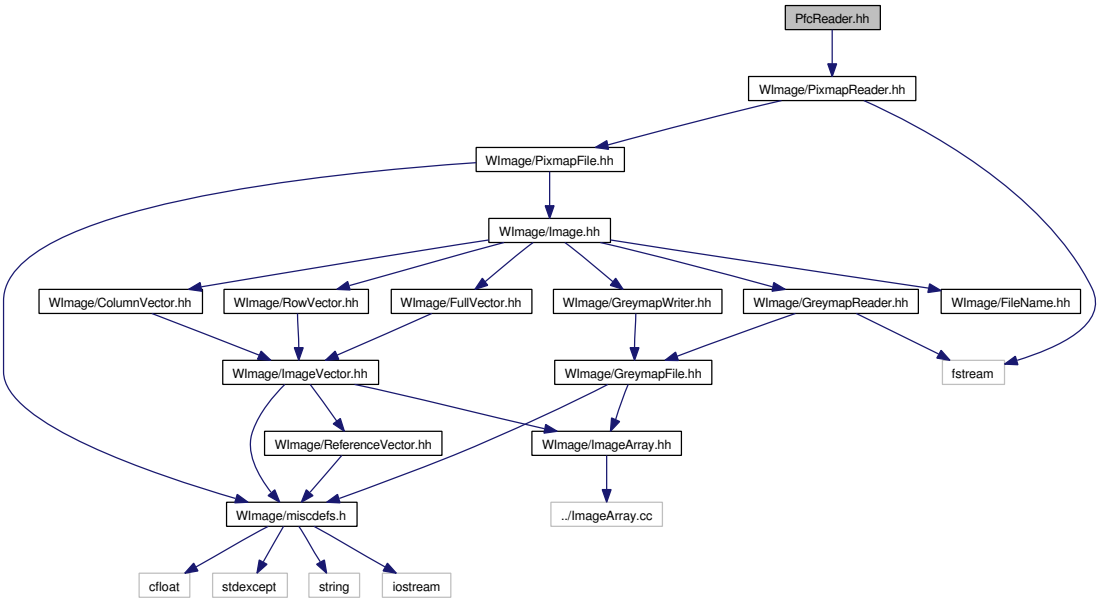
Classes

- class PeriodicPosition

9.31 PfcReader.hh File Reference

`#include "WImage/PixmapReader.hh"`

Include dependency graph for PfcReader.hh:



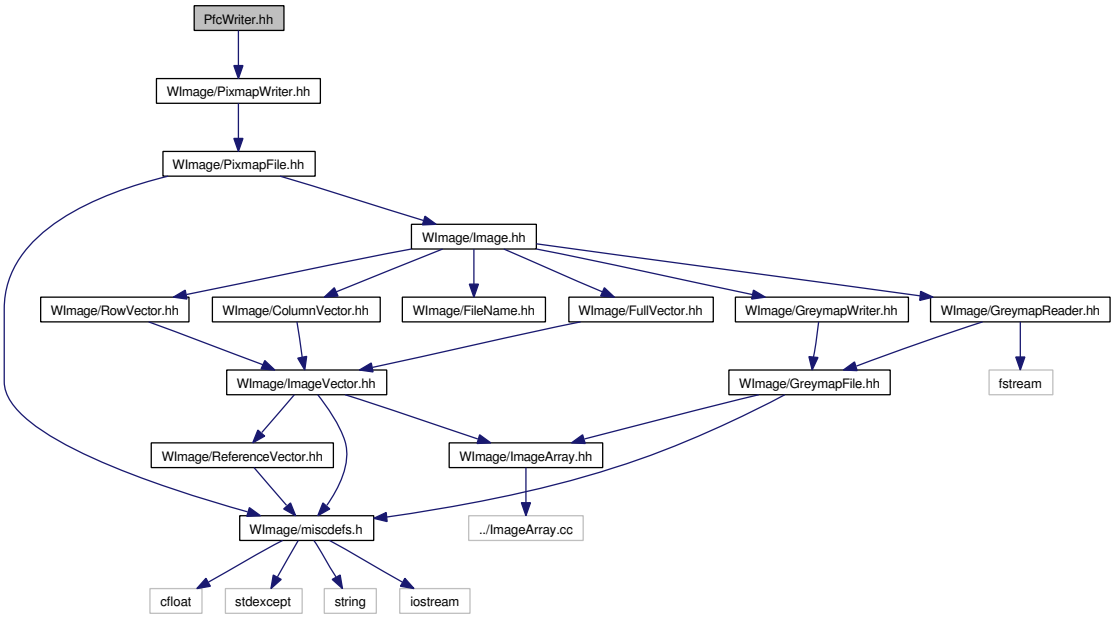
Classes

- class PfcReader

9.32 PfcWriter.hh File Reference

```
#include "WImage/PixmapWriter.hh"
```

Include dependency graph for PfcWriter.hh:



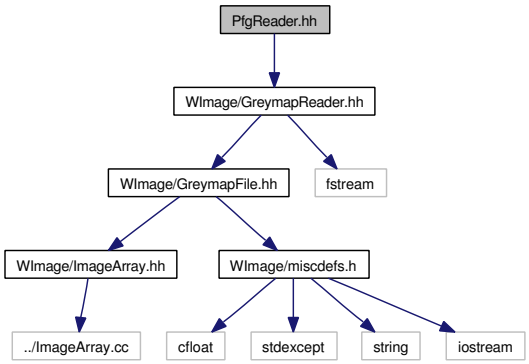
Classes

- class PfcWriter

9.33 PfgReader.hh File Reference

```
#include "WImage/GreymapReader.hh"
```

Include dependency graph for PfgReader.hh:



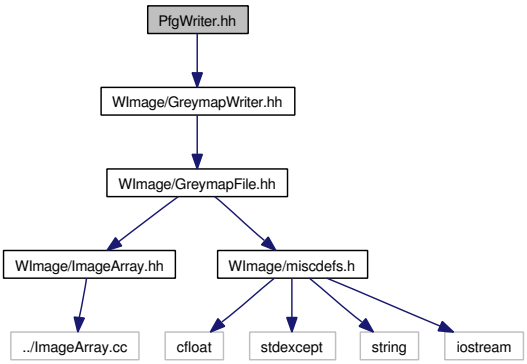
Classes

- class PfgReader

9.34 PfgWriter.hh File Reference

```
#include "WImage/GreymapWriter.hh"
```

Include dependency graph for PfgWriter.hh:



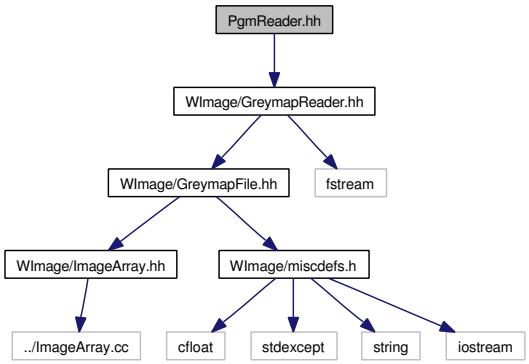
Classes

- class PfgWriter

9.35 PgmReader.hh File Reference

```
#include "WImage/GreymapReader.hh"
```

Include dependency graph for PgmReader.hh:



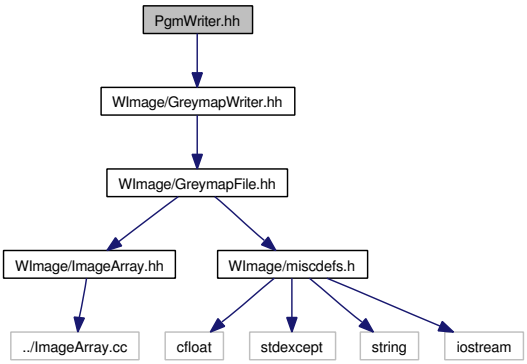
Classes

- class PgmReader

9.36 PgmWriter.hh File Reference

```
#include "WImage/GreymapWriter.hh"
```

Include dependency graph for PgmWriter.hh:



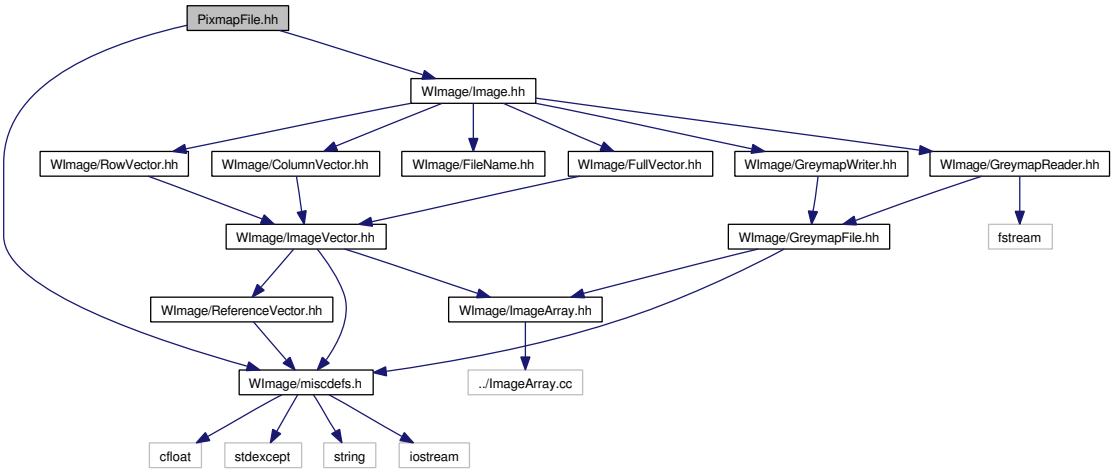
Classes

- class PgmWriter

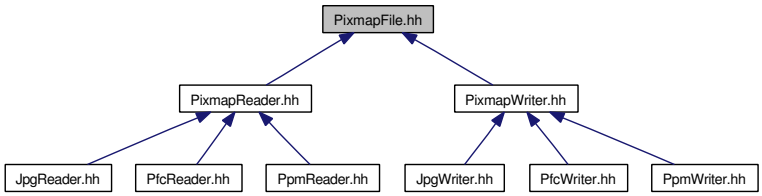
9.37 PixmapFile.hh File Reference

```
#include "WImage/Image.hh"
#include "WImage/miscdefs.h"
```

Include dependency graph for PixmapFile.hh:



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



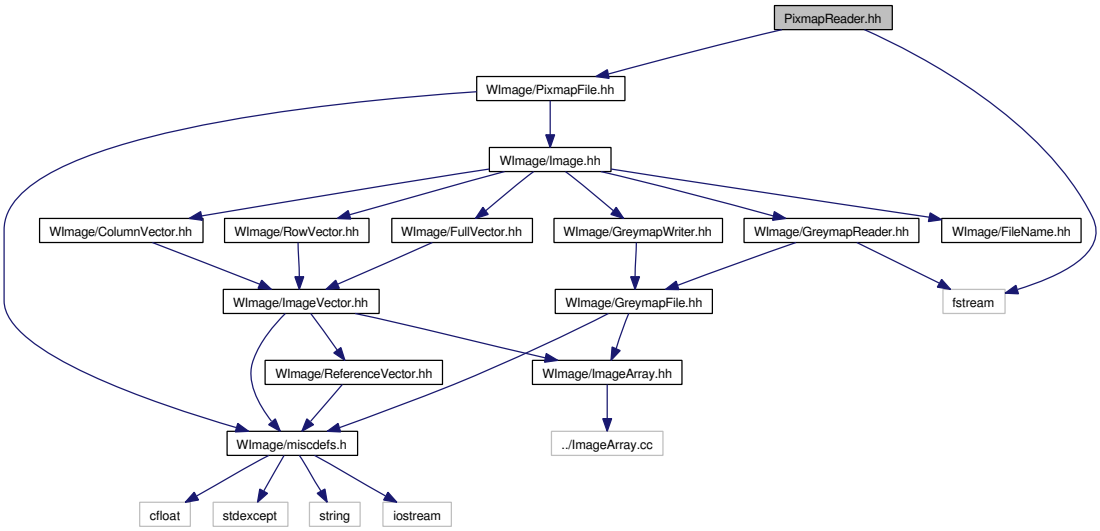
Classes

- class PixmapFile

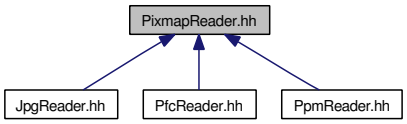
9.38 PixmapReader.hh File Reference

```
#include "WImage/PixmapFile.hh"
#include <fstream>
```

Include dependency graph for PixmapReader.hh:



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



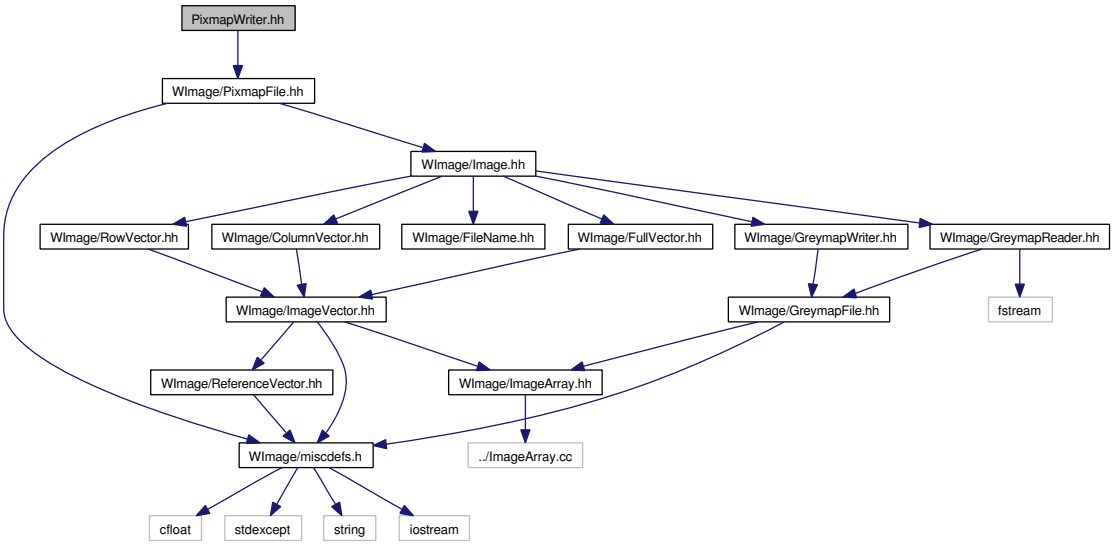
Classes

- class PixmapReader

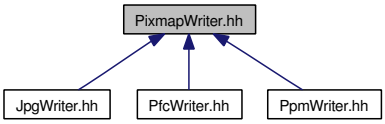
9.39 PixmapWriter.hh File Reference

`#include "WImage/PixmapFile.hh"`

Include dependency graph for PixmapWriter.hh:



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



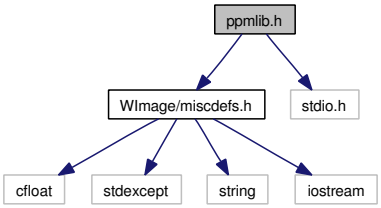
Classes

- class PixmapWriter

9.40 ppmlib.h File Reference

```
#include "WImage/miscdefs.h"
#include <stdio.h>
```

Include dependency graph for ppmlib.h:



Functions

- pixel * ppm_read (const char *fname, int *height, int *width, int *cmax)
- int ppm_write (const char *fname, pixel *pic, int height, int width, int cmax)
- pixel * ppm_fromStream (FILE *in, int *height, int *width, int *cmax)
- int ppm_toStream (FILE *out, pixel *pic, int height, int width, int cmax)

9.40.1 Function Documentation

9.40.1.1 pixel* ppm_fromStream (FILE * *in*, int * *height*, int * *width*, int * *cmax*)

Read a PPM file from an open stream.

Parameters:

in the open input stream

height the number of pixel rows, returned by the function

width the number of pixel cols, returned by the function

cmax the number of colors, returned by the function

Returns:

an array containing the pixels as RGB triples if successful else NULL

9.40.1.2 int ppm_toStream (FILE * *out*, pixel * *pic*, int *height*, int *width*, int *cmax*)

Write a PPM file to an open stream

Parameters:

out the open output stream

pic the image as an array of RGB triples

height the number of pixel rows

width the number of pixel cols

cmax the maximum color value

Returns:

0 if successful else -1

9.40.1.3 int ppm_write (const char * *fname*, pixel * *pic*, int *height*, int *width*, int *cmax*)

Write a PPM file to the filesystem

Parameters:

fname the filename

pic the image as an array of RGB triples

height the number of pixel rows

width the number of pixel cols

cmax the maximum color value

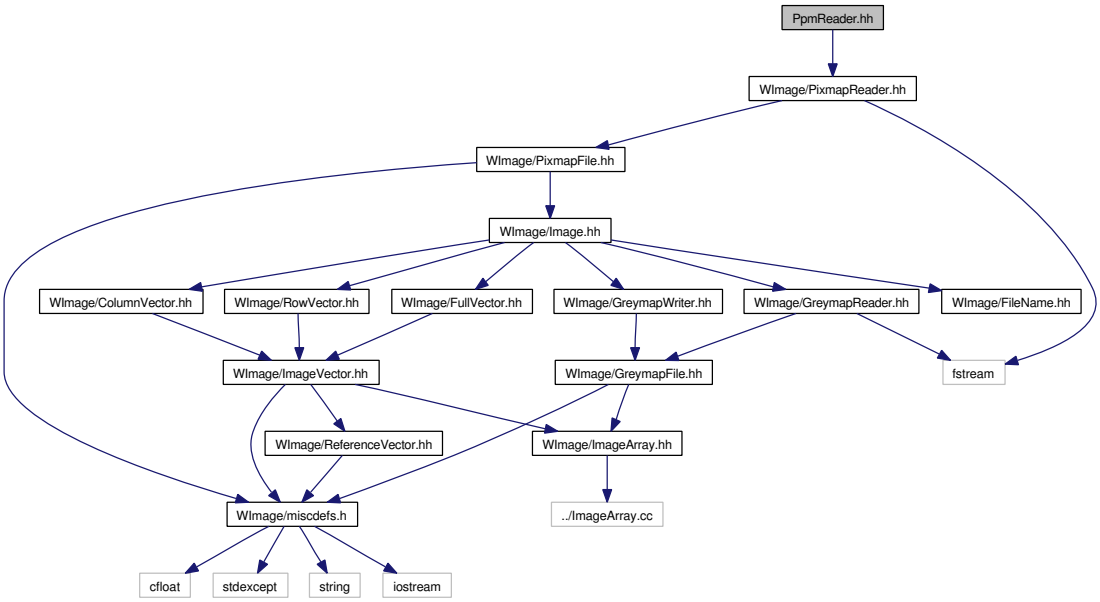
Returns:

0 if successful else -1

9.41 PpmReader.hh File Reference

```
#include "WImage/PixmapReader.hh"
```

Include dependency graph for PpmReader.hh:



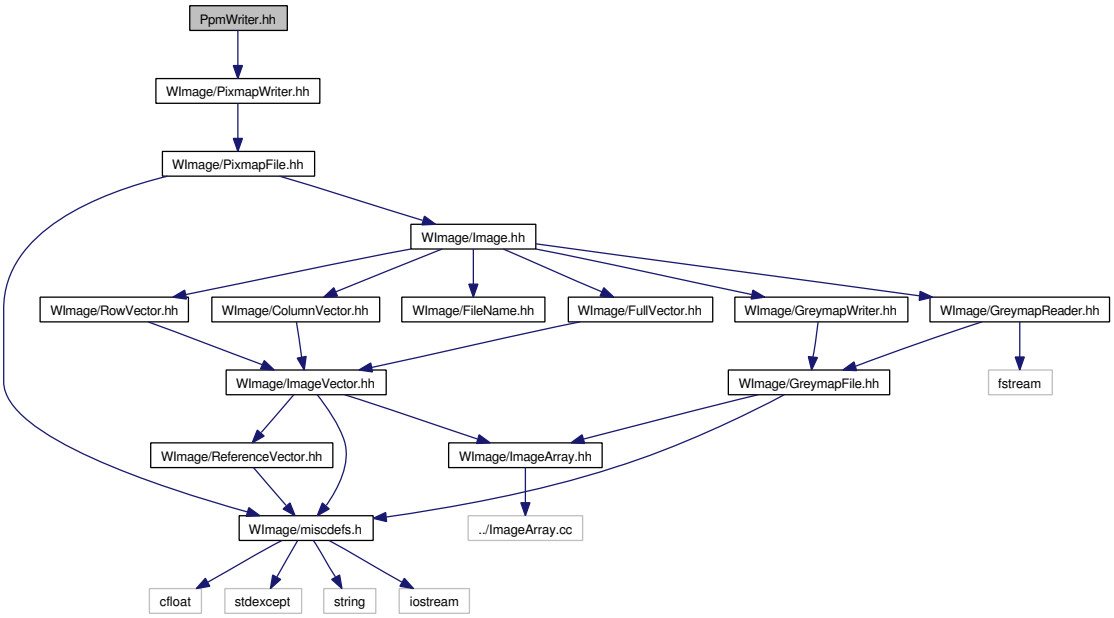
Classes

- class PpmReader

9.42 PpmWriter.hh File Reference

```
#include "WImage/PixmapWriter.hh"
```

Include dependency graph for PpmWriter.hh:



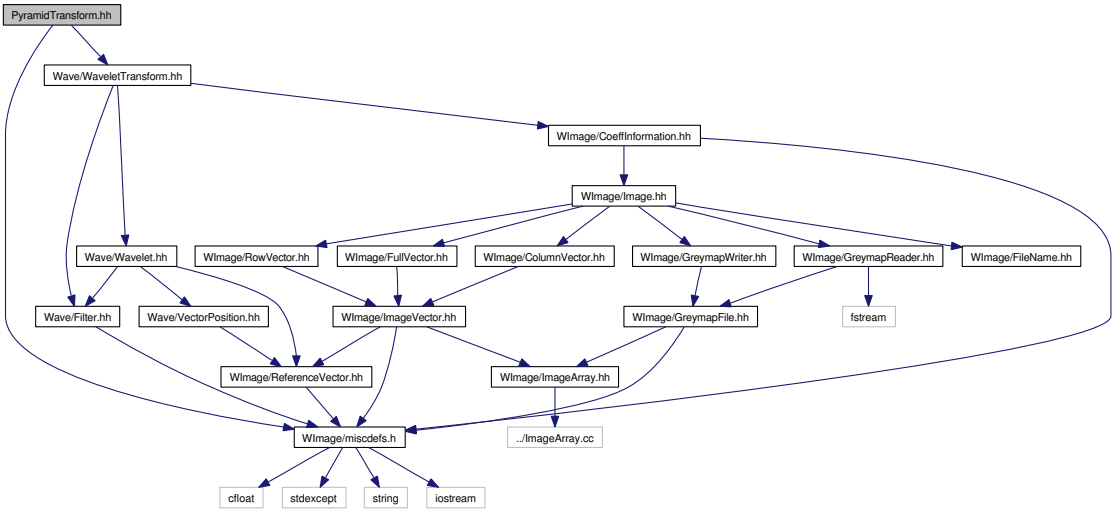
Classes

- class PpmWriter

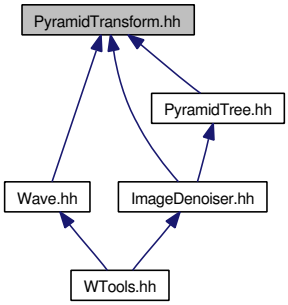
9.43 PyramidTransform.hh File Reference

```
#include "Wave/WaveletTransform.hh"
#include "WImage/miscdefs.h"
```

Include dependency graph for PyramidTransform.hh:



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



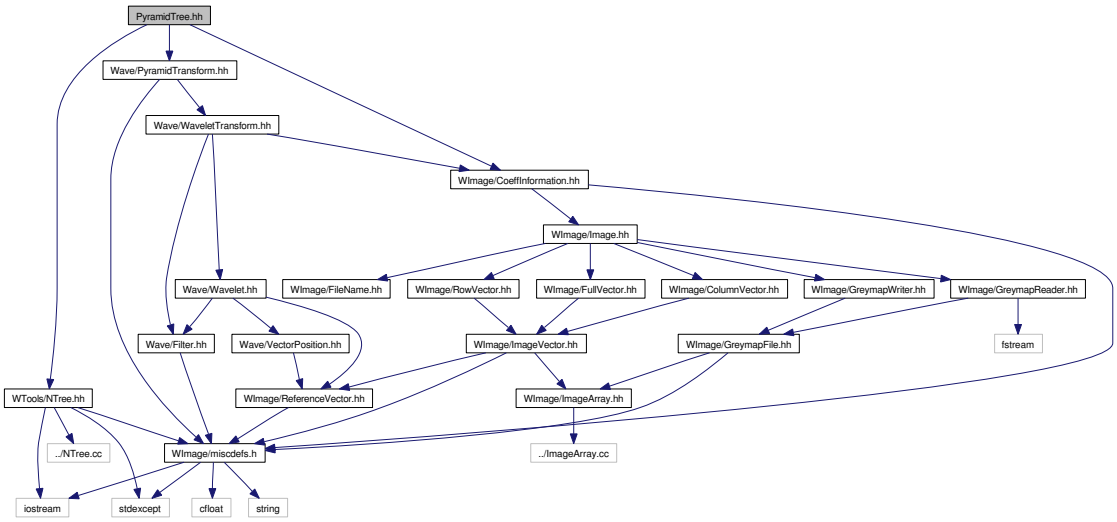
Classes

- class PyramidTransform

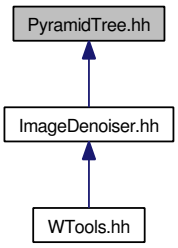
9.44 PyramidTree.hh File Reference

```
#include "WTools/NTree.hh"
#include "WImage/CoeffInformation.hh"
#include "Wave/PyramidTransform.hh"
```

Include dependency graph for PyramidTree.hh:



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



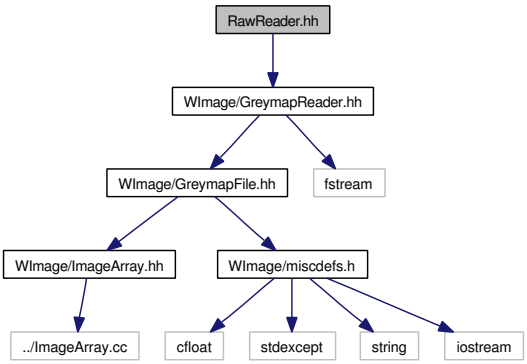
Classes

- class PyramidTree

9.45 RawReader.hh File Reference

```
#include "WImage/GreymapReader.hh"
```

Include dependency graph for RawReader.hh:



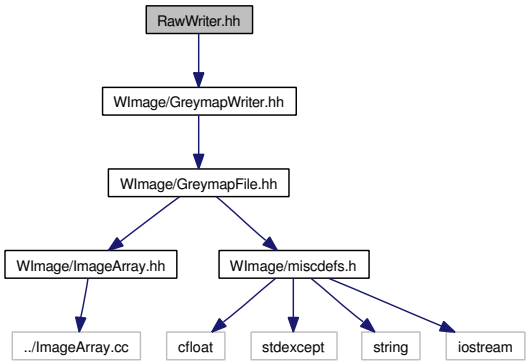
Classes

- class RawReader

9.46 RawWriter.hh File Reference

```
#include "WImage/GreymapWriter.hh"
```

Include dependency graph for RawWriter.hh:



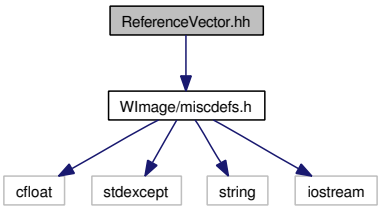
Classes

- class RawWriter

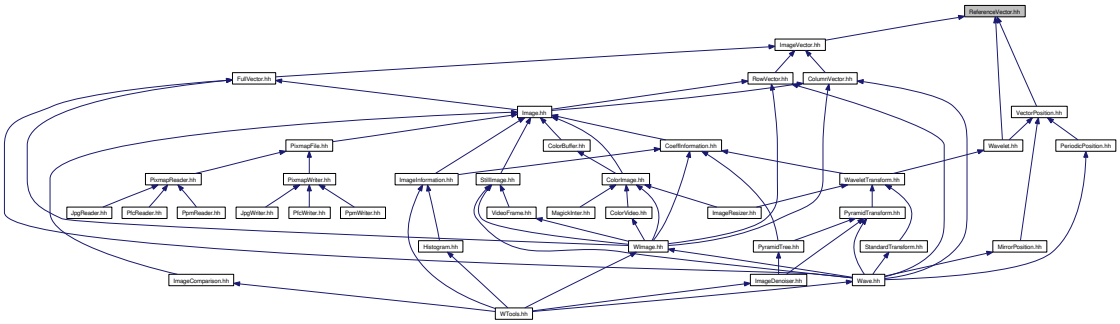
9.47 ReferenceVector.hh File Reference

```
#include "WImage/miscdefs.h"
```

Include dependency graph for ReferenceVector.hh:



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



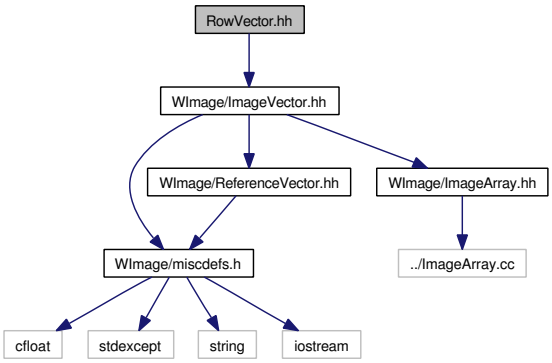
Classes

- class ReferenceVector

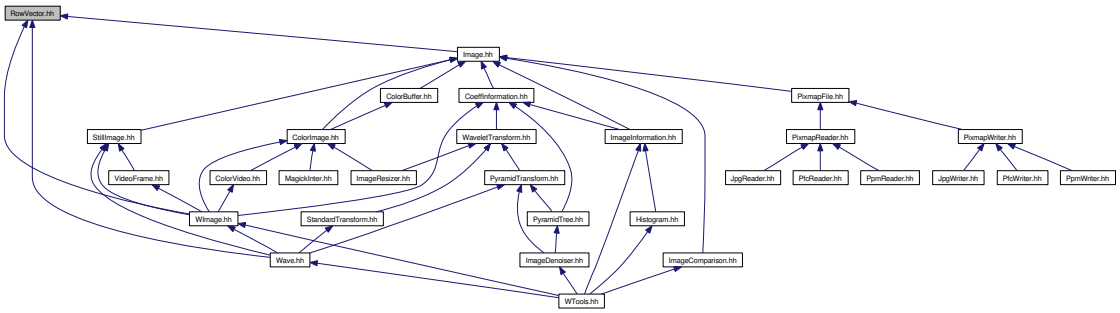
9.48 RowVector.hh File Reference

`#include "WImage/ImageVector.hh"`

Include dependency graph for RowVector.hh:



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



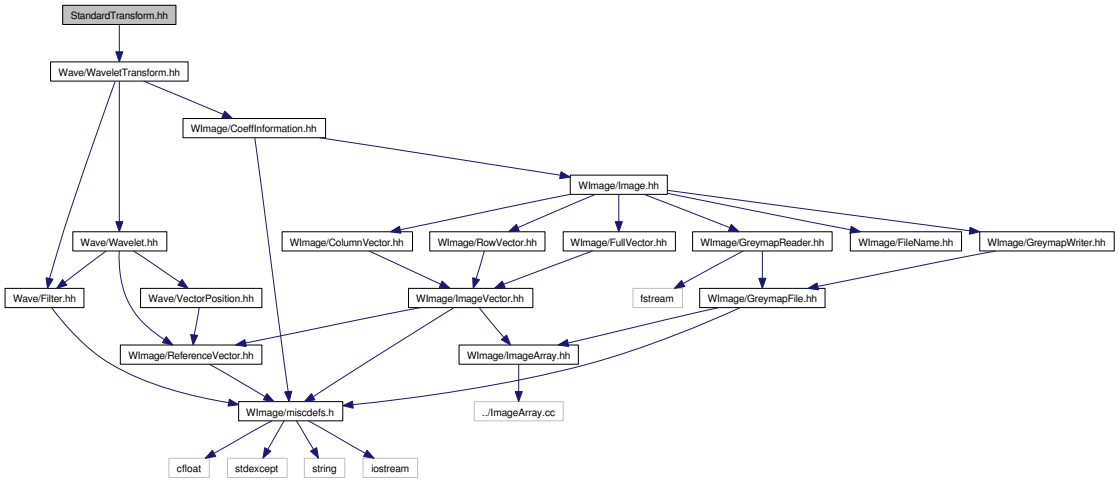
Classes

- class RowVector

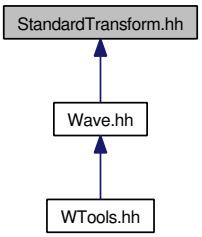
9.49 StandardTransform.hh File Reference

```
#include "Wave/WaveletTransform.hh"
```

Include dependency graph for StandardTransform.hh:



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



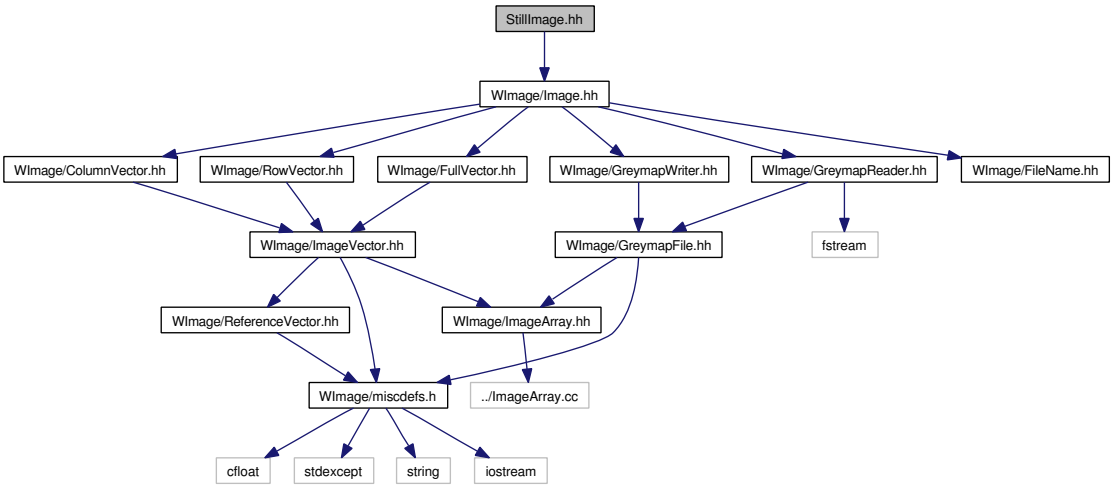
Classes

- class StandardTransform

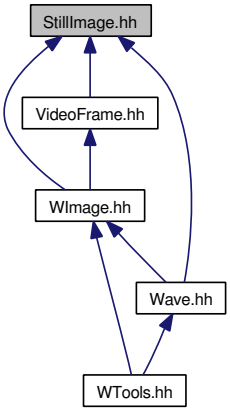
9.50 StillImage.hh File Reference

```
#include "WImage/Image.hh"
```

Include dependency graph for StillImage.hh:



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



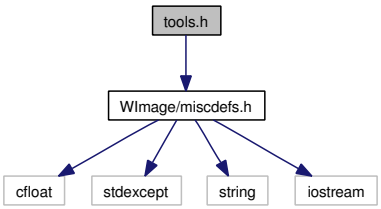
Classes

- class StillImage

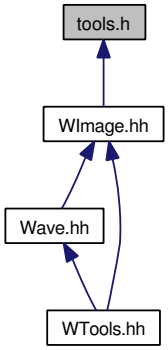
9.51 tools.h File Reference

```
#include "WImage/miscdefs.h"
```

Include dependency graph for tools.h:



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



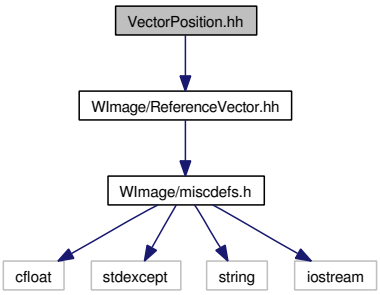
Functions

- pixel tools_coeff2pixel (coeff c)
- int tools_coeff2int (coeff c)
- bool tools_equals (double d1, double d2)
- bool tools_epsilons (double d1, double d2, double epsilon)
- bool tools_areaFromString (char a1, char a2, area &result)
- bool tools_areaFromString (char *a, area &result)
- const char * tools_areaToString (const area a)
- bool tools_powerOfTwo (int value, int &power)
- int tools_startFromCenter (int pos, int length)
- int tools_fileSize (const char *fname)

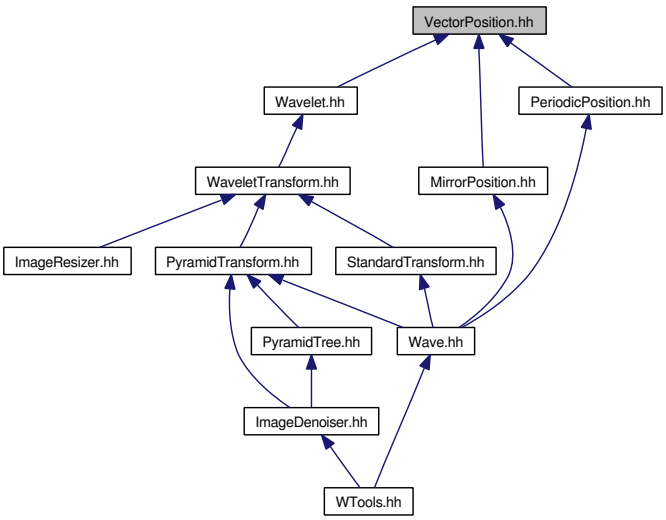
9.52 VectorPosition.hh File Reference

```
#include "WImage/ReferenceVector.hh"
```

Include dependency graph for VectorPosition.hh:



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



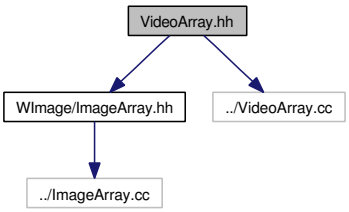
Classes

- class VectorPosition

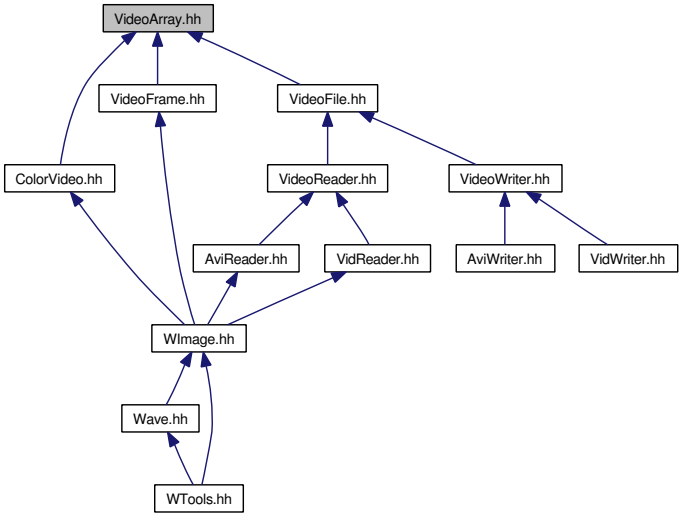
9.53 VideoArray.hh File Reference

```
#include "WImage/ImageArray.hh"
#include "../VideoArray.cc"
```

Include dependency graph for VideoArray.hh:



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



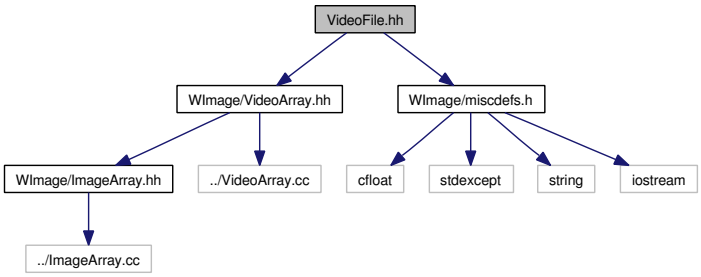
Classes

- class VideoArray< T >

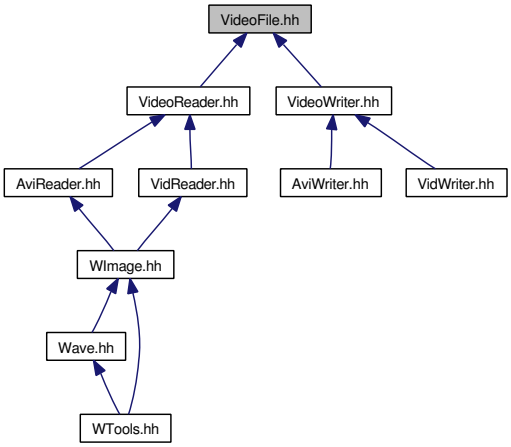
9.54 VideoFile.hh File Reference

```
#include "WImage/VideoArray.hh"
#include "WImage/miscdefs.h"
```

Include dependency graph for VideoFile.hh:



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



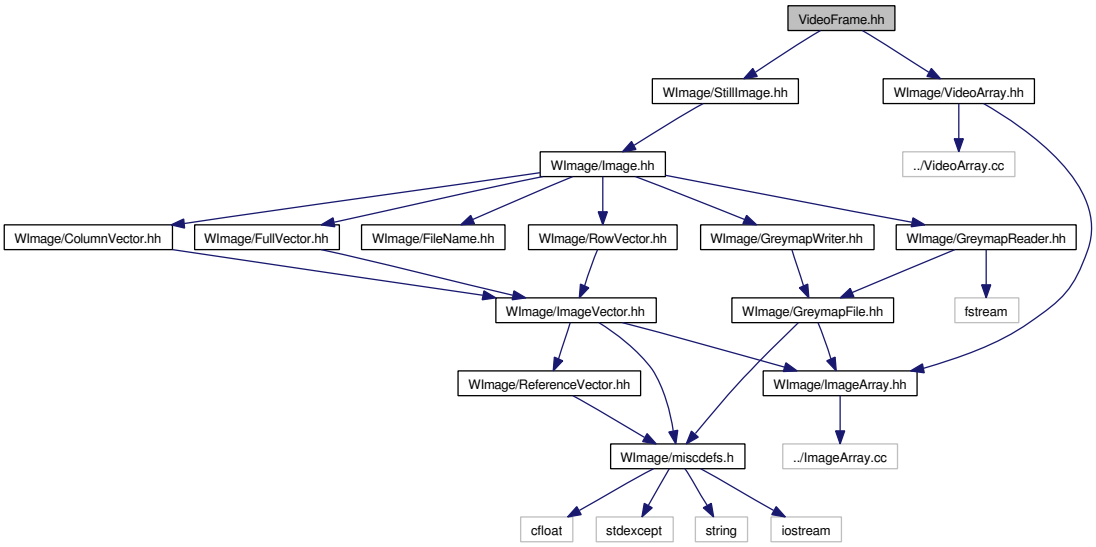
Classes

- class VideoFile

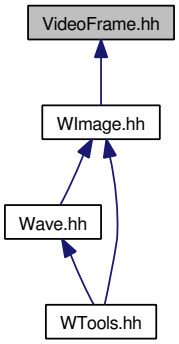
9.55 VideoFrame.hh File Reference

```
#include "WImage/StillImage.hh"
#include "WImage/VideoArray.hh"
```

Include dependency graph for VideoFrame.hh:



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



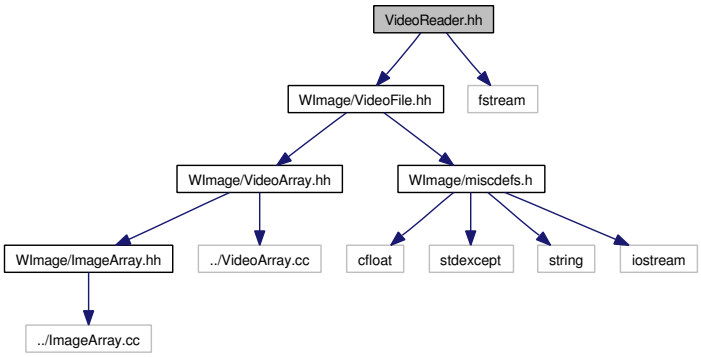
Classes

- class VideoFrame

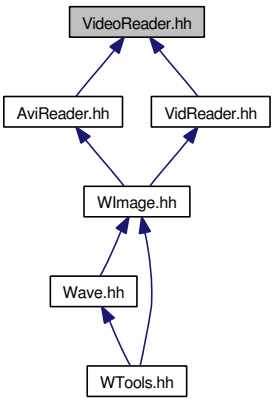
9.56 VideoReader.hh File Reference

```
#include "WImage/VideoFile.hh"
#include <fstream>
```

Include dependency graph for VideoReader.hh:



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



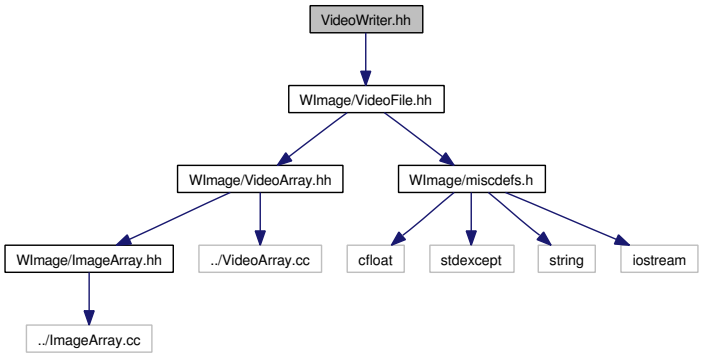
Classes

- class VideoReader

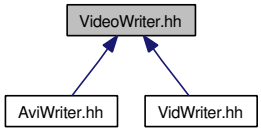
9.57 VideoWriter.hh File Reference

```
#include "WImage/VideoFile.hh"
```

Include dependency graph for VideoWriter.hh:



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



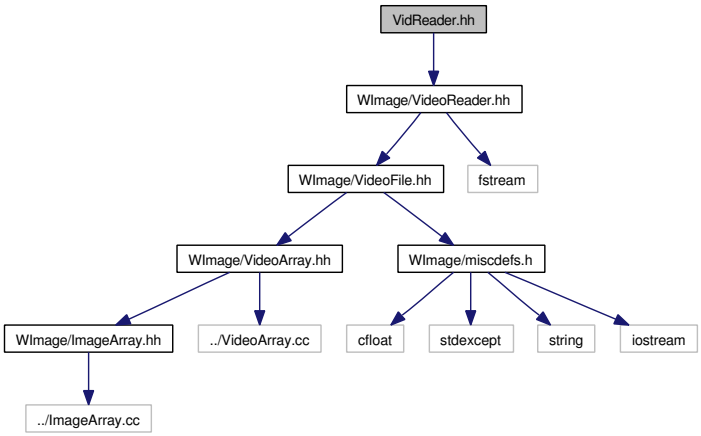
Classes

- class VideoWriter

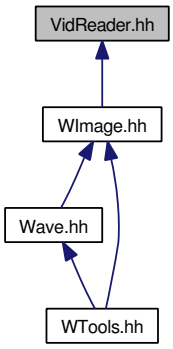
9.58 VidReader.hh File Reference

```
#include "WImage/VideoReader.hh"
```

Include dependency graph for VidReader.hh:



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



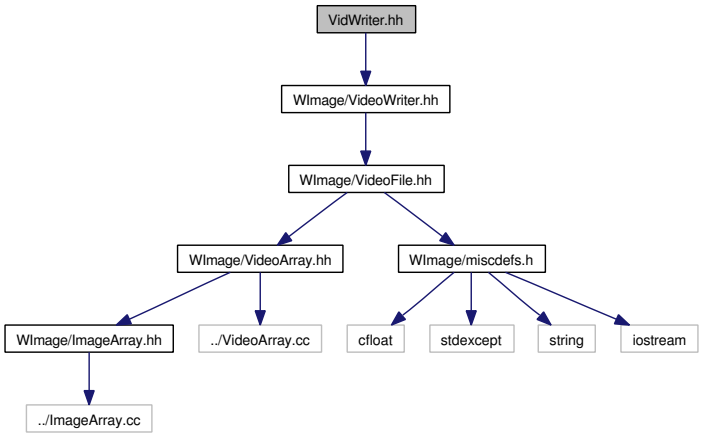
Classes

- class VidReader

9.59 VidWriter.hh File Reference

```
#include "WImage/VideoWriter.hh"
```

Include dependency graph for VidWriter.hh:



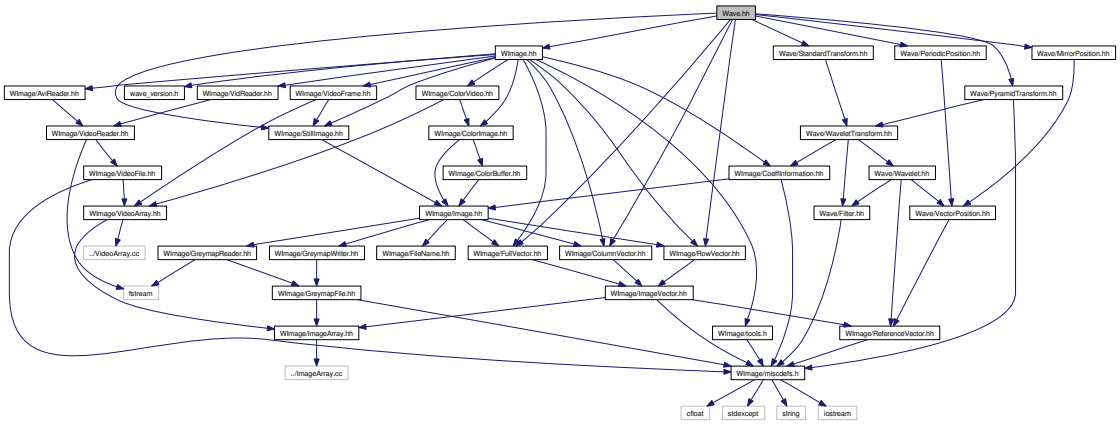
Classes

- class VidWriter

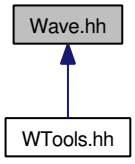
9.60 Wave.hh File Reference

```
#include "WImage.hh"
#include "WImage/RowVector.hh"
#include "WImage/ColumnVector.hh"
#include "WImage/FullVector.hh"
#include "WImage/StillImage.hh"
#include "Wave/StandardTransform.hh"
#include "Wave/PyramidTransform.hh"
#include "Wave/MirrorPosition.hh"
#include "Wave/PeriodicPosition.hh"
```

Include dependency graph for Wave.hh:

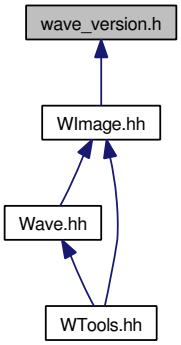


This graph shows which files directly or indirectly include this file:



9.61 wave_version.h File Reference

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



Defines

- #define WAVE_VERSION "1.2-cvs"

9.61.1 Define Documentation

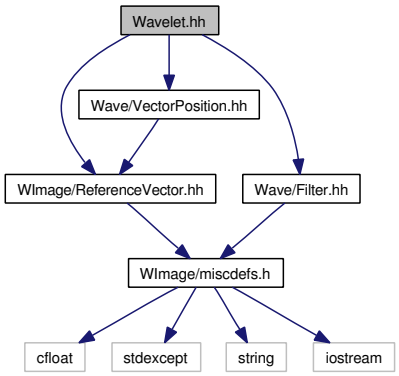
9.61.1.1 #define WAVE_VERSION "1.2-cvs"

Definition at line 14 of file wave_version.h.

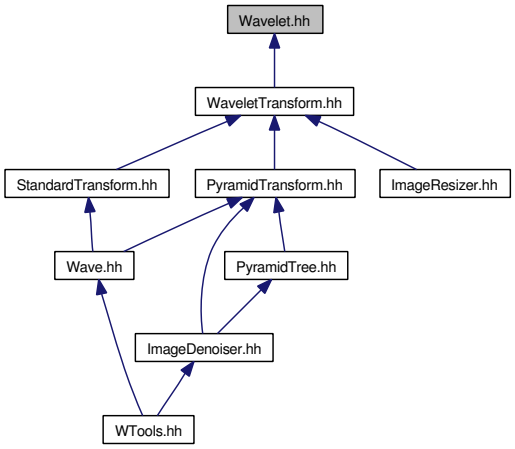
9.62 Wavelet.hh File Reference

```
#include "WImage/ReferenceVector.hh"
#include "Wave/VectorPosition.hh"
#include "Wave/Filter.hh"
```

Include dependency graph for Wavelet.hh:



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



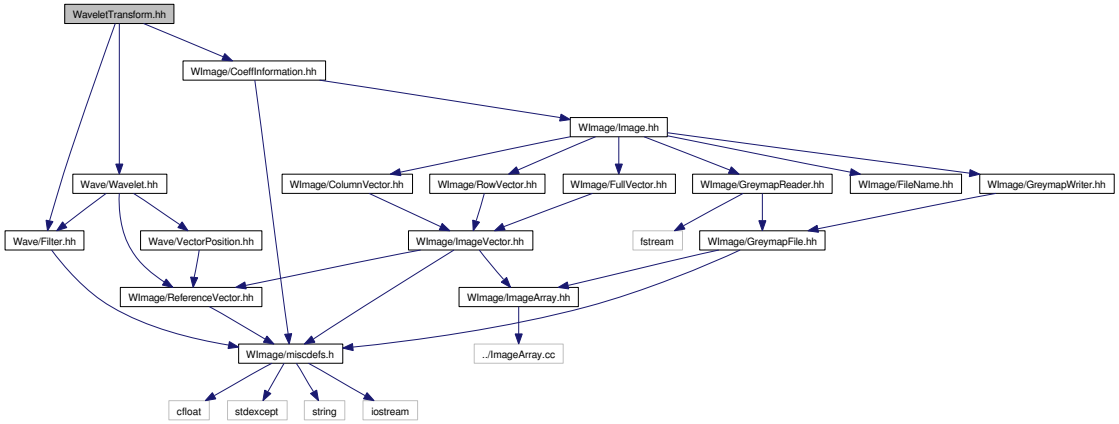
Classes

- class Wavelet

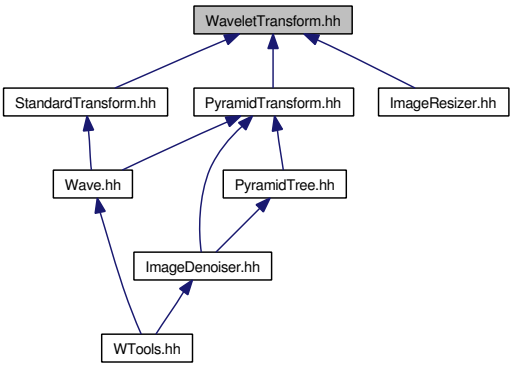
9.63 WaveletTransform.hh File Reference

```
#include "Wave/Wavelet.hh"
#include "WImage/CoeffInformation.hh"
#include "Wave/Filter.hh"
```

Include dependency graph for WaveletTransform.hh:



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



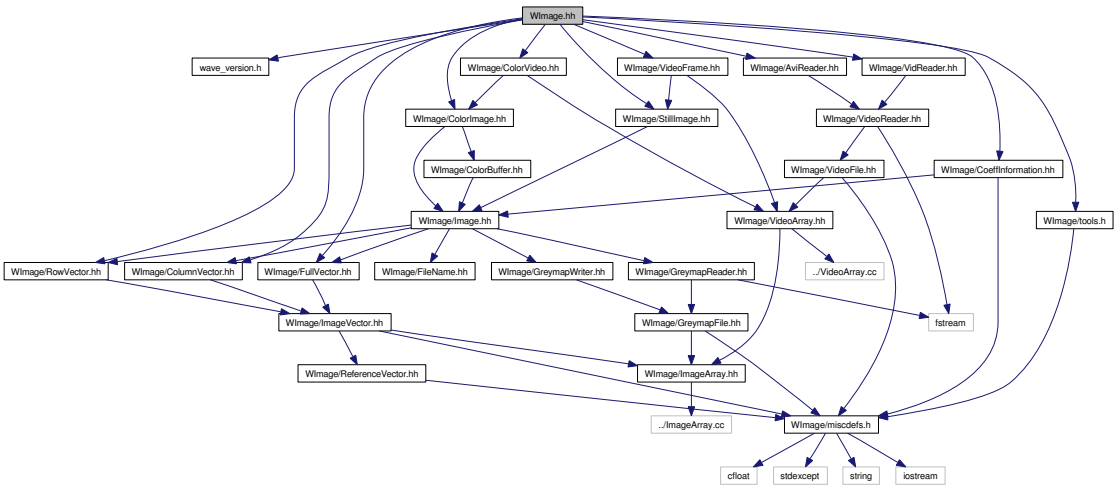
Classes

- class WaveletTransform

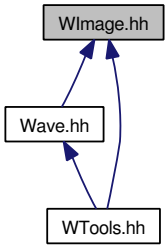
9.64 WImage.hh File Reference

```
#include "wave_version.h"
#include "WImage/RowVector.hh"
#include "WImage/ColumnVector.hh"
#include "WImage/FullVector.hh"
#include "WImage/StillImage.hh"
#include "WImage/ColorImage.hh"
#include "WImage/ColorVideo.hh"
#include "WImage/VideoFrame.hh"
#include "WImage/AviReader.hh"
#include "WImage/VidReader.hh"
#include "WImage/CoeffInformation.hh"
#include "WImage/tools.h"
```

Include dependency graph for WImage.hh:



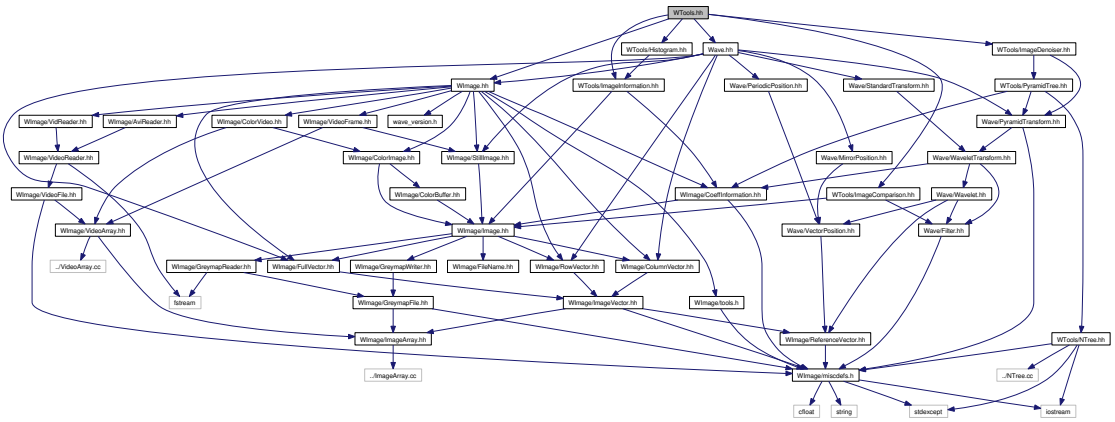
This graph shows which files directly or indirectly include this file:



9.65 WTools.hh File Reference

```
#include "WImage.hh"
#include "Wave.hh"
#include "WTools/ImageInformation.hh"
#include "WTools/ImageComparison.hh"
#include "WTools/Histogram.hh"
#include "WTools/ImageDenoiser.hh"
```

Include dependency graph for WTools.hh:



Index