# CDVS Library

14.0

Generated by Doxygen 1.8.6

Thu Jun 11 2015 11:17:42

# Contents

# Chapter 1

# Documentation

## 1.1 Introduction

This is the documentation of the CDVS Library, a library with minimal dependencies to encode/decode and use MPEG-7 compliant Compact Descriptors for Visual Search. The binary format of CDVS descriptors is specified in ISO/IEC FDIS 15938-13 "Compact Descriptors for Visual Search". For further information see http://mpeg.-chiariglione.org/standards/mpeg-7/compact-descriptors-visual-search

The CDVS Library structure is based on:

- an abstract API using the namespace “mpeg7cdvs”;
- a set of libraries composed by three implementation variants:
  - libcdvs_main: the reference implementation;
  - libcdvs_lowmem: a low memory implementation of the ALP keypoint detector;
  - libcdvs_bflog: an alternative implementation of the ALP keypoint detector based on fast fourier transforms (needs the libfftw3f library);

## 1.2 Abstract API

The CDVS abstract API is composed by three abstract base classes and a factory method to get an actual implementation of the classes; the implementation is composed by derived classes that implement the virtual methods declared in the abstract base classes.

The CDVS abstract API is contained in a file named CdvsInterface.h and contains the following classes:

- CdvsConfiguration - interface to all configuration parameters for clients and servers;
- CdvsClient - interface to the client-side functionality of the CDVS Library;
- CdvsServer - interface to the server-side functionality of the CDVS Library;

All classes have a factory method and virtual abstract methods that define the API. In the following we provide a brief description, whereas a complete reference documentation of the API is available in this document.

### 1.2.1 CdvsConfiguration

```
class CdvsConfiguration {
public:
    virtual ~CdvsConfiguration() {};
    static CdvsConfiguration * cdvsConfigurationFactory(const char * configfile = NULL);
```

```
    virtual const Parameters & getParameters(int mode) const = 0;
    virtual Parameters & setParameters(int mode) = 0;
    static int getMode(int descLen);
};
```

CdvsConfiguration contains a static factory method “cdvsConfigurationFactory” that produces an instance of itself; if no parameter is given the configuration will contain the correct default parameter values for all modes from 0 to 6. If a “configfile” is passed as parameter, the factory method will read the file and change the default values accondingly; the format of the file must be the following:

```
[Mode = n]
paramName = value
```

Example:

```
[Mode = 1]
wmThreshold2Way = 1.64
gdThreshold = 9.285

[Mode = 2]
wmThreshold = 2.7
wmMixed = 2.785

[Mode = 3]
wmThreshold = 2.12
wmMixed = 2.19
gdThresholdMixed = 6.025

[Mode = 4]
gdThreshold = 7.235

[Mode = 5]
wmThreshold = 2.165

[Mode = 6]
wmThreshold = 2.15
```

The list of parameters that can be changed is the following:

```
int descLength;                          length in bytes of the CDVS descriptor (i.e. 512, 1024, 2048)
int resizeMaxSize;                       maximum size of one side of the image
int blockWidth;                          coordinate coding: spatial resolution of the coordinates (max
int ctxTableIdx;                         coordinate coding: index of the context table to use
char modeExt[40];                        descriptor extension
unsigned int selectMaxPoints;            feature extraction: max number of points used to describe an
unsigned int numRelevantPoints;          feature extraction: number of points considered relevant in t
float ratioThreshold;                    DISTRAT: threshold for descriptor matching
unsigned int minNumInliers;              DISTRAT: min number of inliers after the geometric check
double wmThreshold;                      Weighted matching threshold
double wmThreshold2Way;                  Two way matching weighted threshold
double wmMixed;                          Weighted matching threshold for mixed cases
double wmMixed2Way;                      Two way weighted matching threshold for mixed cases
int debugLevel;                          0 = off, 1 = on (quiet), 2 = on (verbose), 3 = verbose + dump
int ransacNumTests;                      RANSAC: number of iterations in RANSAC
float ransacThreshold;                   RANSAC: distortion threshold to be used by RANSAC
unsigned int chiSquarePercentile;        percentile used in DISTRAT for Chi-square computation
int retrievalLoops;                      number of loops performed in the final stage of the retrieval
double wmRetrieval;                      Weighted matching threshold for retrieval
double wmRetrieval2Way;                  Two way weighted matching threshold for retrieval
int retrievalMaxPoints;                  max number of points used in the retrieval experiment
int queryExpansionLoops;                 number of query expansion loops to perform in the retrieval e
float scfvThreshold;                     threshold value to control the sparsity of scfv vector
bool hasVar;                             indicates if using the gradient vector w.r.t the variance of
float locationBits;                      average bits per key point to encode location information;
bool hasBitSelection;                    indicates if the Global Descriptor uses the bit selection alg
float gdThreshold;                       global descriptor threshold
float gdThresholdMixed;                  global descriptor threshold for mixed cases
```

The getParameter and setParameter methods allow to respectively read and modify some of the parameters stored in the CdvsConfiguration instance from the application code, instead of using a text file. Finally, getMode is just a convenience method to get a mode ID corresponding to a descriptor length, to help developers to select the appropriate mode for their application.

For a complete documentation of this class, see mpeg7cdvs::CdvsConfiguration

### 1.2.2 CdvsClient

```
class CdvsClient {
public:
    virtual ~CdvsClient() {};
    static CdvsClient * cdvsClientFactory(const CdvsConfiguration * config, int mode);
    virtual unsigned int encode(CdvsDescriptor & output, int width, int height, const unsigned char * input) c
};
```

The CdvsClient class has a factory method named “cdvsClientFactory” which requires a CdvsConfiguration instance and a mode as input parameters. The only other method of the class is the “encode” method which encodes the luminance component of an image producing a CDVS descriptor. The encode method can be called multiple times to produce a set of descriptors, all of which will be produced using the parameter set associated to the same mode (the mode that was specified in the factory method). The encode method is declared “const” to indicate that it does not modify any internal variable of the CdvsClient instance, therefore it can be called safely in multithreading mode, provided that the CdvsDescriptor output parameter (which is modified!) is declared in the address space of the thread (and not shared).

For a complete documentation of this class, see mpeg7cdvs::CdvsClient

### 1.2.3 CdvsServer

```
class CdvsServer {
public:
    virtual ~CdvsServer() {};
    static CdvsServer * cdvsServerFactory(const CdvsConfiguration * config, bool twoWayMatch = true);
    virtual size_t decode(CdvsDescriptor & output, const char * fname) const = 0;
    virtual size_t decode(CdvsDescriptor & output, const unsigned char * bitstream = NULL, int size = 0) const
    virtual PointPairs match(const CdvsDescriptor & queryDescriptor, const CdvsDescriptor & refDescriptor, con
    virtual PointPairs match(const CdvsDescriptor & queryDescriptor, unsigned int index, const CDVSPOINT *r_bb
    virtual void createDB(int mode, int reserve) = 0;
    virtual unsigned int addDescriptorToDB(const CdvsDescriptor & refDescriptor, const char * referenceImageId
    virtual bool isDescriptorInDB(const char * referenceImageId) const = 0;
    virtual bool replaceDescriptorInDB(const CdvsDescriptor & refDescriptor, const char * referenceImageId, co
    virtual void clearDB() = 0;
    virtual void commitDB() = 0;
    virtual void storeDB(const char * localname, const char * globalname) const = 0;
    virtual void loadDB(const char * localname, const char * globalname) = 0;
    virtual size_t sizeofDB() const = 0;
    virtual int retrieve(std::vector<RetrievalData> & results, const CdvsDescriptor & queryDescriptor, unsigne
    virtual std::string getImageId(unsigned int index) const = 0;
};
```

The CdvsServer class has a factory method named cdvsServerFactory which requires a CdvsConfiguration instance and an optional boolean value indicating whether to use two way matching or not. This class has more abstract methods than the other two because it incorporates all the functionality required by a server to decode CDVS descriptors, use a pair of them for matching, aggregate many of them in a database, and finally use a descriptor as a query in a retrieval operation. It can also load a DB from a file, or store a DB into a file, making it permanent.

Also in this case, some methods are marked as “const” to indicate that they do not modify any internal variable of the CdvsServer instance and as such, they can be called safely in a multithreading section of the application code. Methods not marked as “const” like createDB and loadDB cannot be called from a multithreading application as they are not thread-safe.

For a complete documentation of this class, see mpeg7cdvs::CdvsServer

## 1.3 The libraries

The CDVS Library code is entirely written in standard C and C++, and can be compiled on 32 or 64-bit environments (mainly Windows and Linux). The library can be easily installed on Linux using autotools. Instruction are given in the README and INSTALL files. The default installation on Linux will place the library in the /usr/local directory, in particular all header files will be placed in /usr/local/include and all libraries in /usr/local/lib. The makefile produces three variants of the CDVS library, each one containing a specific ALP Detector implementation (main, low memory, bflog):

```
libcdvs_bflog.a   libcdvs_lowmem.a   libcdvs_main.a
libcdvs_bflog.la  libcdvs_lowmem.la  libcdvs_main.la
libcdvs_bflog.so  libcdvs_lowmem.so  libcdvs_main.so
```

An application using the CDVS library can select to use any of the three detectors without changing a single line of code, by simply linking the desired library. Moreover, each library is produced in a static and a dynamic version so that application developers are free to use the solution they prefer.

## 1.4 Examples of usage

The CDVS Library is very simple to use. A calling application must include just one header file, and link against one of the three implementations of the CDVS Library. For example, to encode a CDVS descriptor:

```
#include "CdvsInterface.h"
using namespace mpeg7cdvs;

int main(int argc, char *argv[])
{
    int mode = 4;

    // use default configuration (as specified in CDVS)
    CdvsConfiguration * cdvsconfig = CdvsConfiguration::cdvsConfigurationFactory();
    CdvsClient * cdvsclient = CdvsClient::cdvsClientFactory(cdvsconfig, mode);

    // ... read an image here, get the luminance component, width and height of the image

    CdvsDescriptor output;
    size_t descriptor_length = cdvsclient->encode(output, width, height, luminance);
    output.buffer.write("filename.cdvs");
    delete cdvsconfig;          // destroy the CDVS configuration instance
    delete cdvsclient;          // destroy the CDVS client instance
}
```

A more complete example is given below, which encodes one image and uses it as a query in a retrieval operation.

```
#include <iostream>
#include <string>
#include <jpeglib.h>
#include "CdvsInterface.h"
#include "CdvsException.h"

using namespace mpeg7cdvs;
using namespace std;

class JpegReader {
... // see the JpegReader implementation in extract.cpp
};

int main(int argc, char *argv[])
{
        int modeID = 5;
        cout << "example 2: retrieve imageA.jpg from DB.cdvs using modeID = " << modeID << endl;

        CdvsConfiguration * cdvsconfig = CdvsConfiguration::cdvsConfigurationFactory(); // use default values
        CdvsClient * cdvsclient = CdvsClient::cdvsClientFactory(cdvsconfig, modeID);
```

```
        CdvsServer * cdvsserver = CdvsServer::cdvsServerFactory(cdvsconfig);

        try {
                CdvsDescriptor query;
                vector<RetrievalData> results;
                int max_matches = 40;

                int w,h;
                unsigned char * imageA = JpegReader::readJpeg("imageA.jpg",w,h);
                cdvsclient->encode(query, w,h,imageA);
                delete [] imageA;

                cdvsserver->loadDB("DB.cdvs.local", "DB.cdvs.global");
                int n = cdvsserver->retrieve(results, query, max_matches);

                // Write the results in standard output

                cout << "image ID - local score - global score" << endl;

                for(unsigned int k=0; k<results.size(); ++k)
                {
                  string imgname = cdvsserver->getImageId(results[k].index);
                  cout << imgname << " " << results[k].fScore << " " << results[k].gScore << endl;
                }

        }
        catch(exception & ex)                           // catch any exception, including CdvsException
        {
                cerr << argv[0] << " exception: " << ex.what() << endl;
        }

        delete cdvsclient;
        delete cdvsserver;
        delete cdvsconfig;

        return 0;
}
```

# Chapter 2

# Namespace Index

## 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

# Chapter 3

# Hierarchical Index

## 3.1 Class Hierarchy

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

# Chapter 4

# Data Structure Index

## 4.1 Data Structures

Here are the data structures with brief descriptions:

# Chapter 5

# File Index

## 5.1 File List

Here is a list of all files with brief descriptions:

# Chapter 6

# Namespace Documentation

## 6.1 mpeg7cdvs Namespace Reference

Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.

### Data Structures

- class AbstractDetector

  *Base class for keypoint detectors.*
- class BitInputStream

  *This class represents an input stream of bits.*
- class BitOutputStream

  *This class represents an output stream of bits.*
- class Buffer

  *A container class for a byte array, intended to replace all malloc() and new() instructions in the main code.*
- class CdvsDescriptor

  *Helper class to read/write/check CDVS descriptors according to the syntax defined in ISO/IEC 15938-13.*
- class CdvsException

  *Class defining a specific exception for CDVS.*
- class CdvsConfiguration

  *Interface to all configuration parameters for clients and servers.*
- class CdvsClient

  *Interface to the client-side functionality of the CDVS Library.*
- class CdvsServer

  *Interface to the server-side functionality of the CDVS Library.*
- struct CDVSPOINT

  *A structure containing the x and y coordinate of a point in the image.*
- struct RetrievalData

  *A structure containing the output of a retrieval operation.*
- class Feature

  *Container class for the features of a single point (storing coordinates, scale, orientation, peak and descriptor of a point).*
- class FeatureList

  *Container class for all features of an image.*
- class CompressedFeatureList

  *Container class for all compressed features of an image.*

- class ImageBuffer

  *A container class for a bidimensional image; it's the base class of all keypoint detector classes.*
- class Parameters

  *Container for all encoding/decoding parameters associated to each target bitrate defined by MPEG CDVS.*
- class PointPairs

  *Parameter class, used to pass around matched point coordinates.*
- class LookUpTable

  *A simple look up table implementation, to perform a bit count very quickly.*
- class SCFVSignature

  *Container class for a Scalable Fisher Vector binary signature; allows reading/writing from/to a bitstream, fetching/storing from/into a file, and comparing a signature with another.*
- class SCFVIndex

  *A class to manage an indexed list of SCFV signatures.*
- class SCFVFactory

  *A class to produce SCFV signatures.*

### Typedefs

- typedef Parameters ParameterSet [Parameters::nModes]

### Enumerations

- enum { MATCH_TYPE_DEFAULT = 0, MATCH_TYPE_BOTH = 1, MATCH_TYPE_LOCAL = 2, MATCH_TYPE_GLOBAL = 3 }

  *Type of matching.*
- enum { match_2way_INTERSECTION = 0, match_2way_DISJOINT1 = 1, match_2way_DISJOINT2 = 2 }

### Variables

- static const float gama = 0.3f
- static const int num_bit_selection = 24
- static const int PCASiftLength = 32

  *number of principal components in the centroid space*
- static const int numberCentroids = 512

  *number of centroids of the codebook*

## 6.1.1 Detailed Description

Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.

## 6.1.2 Typedef Documentation

### 6.1.2.1 typedef Parameters mpeg7cdvs::ParameterSet[Parameters::nModes]

## 6.1.3 Enumeration Type Documentation

### 6.1.3.1 anonymous enum

Type of matching.

**Enumerator**

***MATCH_TYPE_DEFAULT*** ignore global if local match

***MATCH_TYPE_BOTH*** compute both local and global matching scores

***MATCH_TYPE_LOCAL*** compute only local matching score

***MATCH_TYPE_GLOBAL*** compute only global matching score

#### 6.1.3.2 anonymous enum

**Enumerator**

***match_2way_INTERSECTION*** indicates 2-direction matching pair

***match_2way_DISJOINT1*** indicates 1-direction matching pair: direction A=>B

***match_2way_DISJOINT2*** indicates 1-direction matching pair: direction B=>A

### 6.1.4 Variable Documentation

#### 6.1.4.1 const float mpeg7cdvs::gama = 0.3f [static]

#### 6.1.4.2 const int mpeg7cdvs::num_bit_selection = 24 [static]

#### 6.1.4.3 const int mpeg7cdvs::numberCentroids = 512 [static]

number of centroids of the codebook

#### 6.1.4.4 const int mpeg7cdvs::PCASiftLength = 32 [static]

number of principal components in the centroid space

# Chapter 7

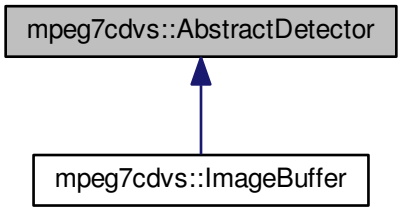
# Data Structure Documentation

## 7.1 mpeg7cdvs::AbstractDetector Class Reference

Base class for keypoint detectors.

`#include <AbstractDetector.h>`

Inheritance diagram for mpeg7cdvs::AbstractDetector:




### Public Member Functions

- AbstractDetector ()
- virtual ~AbstractDetector ()
- virtual void detect (FeatureList &featurelist, const Parameters &params)=0

  *Detect all keypoints from this image.*
- virtual void extract (FeatureList &featurelist, size_t num) const =0

  *Extract the SIFT descriptor of each keypoint and store it back in featurelist.*

### Data Fields

- int height

  *height of the image*
- int width

  *width of the image*
- int originalWidth

*original width of the image (before any resample operation)*

- int originalHeight

  *original height of the image (before any resample operation)*

### 7.1.1 Detailed Description

Base class for keypoint detectors.

**Author**

Massimo Balestri

**Date**

2014

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 mpeg7cdvs::AbstractDetector::AbstractDetector ( ) [inline]

#### 7.1.2.2 virtual mpeg7cdvs::AbstractDetector::~AbstractDetector ( ) [inline], [virtual]

### 7.1.3 Member Function Documentation

#### 7.1.3.1 virtual void mpeg7cdvs::AbstractDetector::detect ( FeatureList & *featurelist,* const Parameters & *params* ) [pure virtual]

Detect all keypoints from this image.

**Parameters**

| | |
|---|---|
| *featurelist* | the ouput list of keypoints with their associated features, in descending order of importance. |
| *params* | the running parameters. |

**Exceptions**

| | |
|---|---|
| *CdvsException* | in case of error |

#### 7.1.3.2 virtual void mpeg7cdvs::AbstractDetector::extract ( FeatureList & *featurelist,* size_t *num* ) const [pure virtual]

Extract the SIFT descriptor of each keypoint and store it back in featurelist.

**Parameters**

| | |
|---|---|
| *featurelist* | the detected keypoints |
| *num* | the absolute maximum number of features to be extracted from this image |

### 7.1.4 Field Documentation

#### 7.1.4.1 int mpeg7cdvs::AbstractDetector::height

height of the image

#### 7.1.4.2 int mpeg7cdvs::AbstractDetector::originalHeight

original height of the image (before any resample operation)

#### 7.1.4.3 int mpeg7cdvs::AbstractDetector::originalWidth

original width of the image (before any resample operation)

#### 7.1.4.4 int mpeg7cdvs::AbstractDetector::width

width of the image

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

- AbstractDetector.h

## 7.2 mpeg7cdvs::BitInputStream Class Reference

This class represents an input stream of bits.

`#include <BitInputStream.h>`

### Public Member Functions

- BitInputStream ()
  *Create an empty object.*
- BitInputStream (const unsigned char *buf, size_t size)
  *Create and initialize a BitInputStream.*
- virtual ~BitInputStream ()
  *Close the stream, if not yet done, and destroy the object.*
- void open (const unsigned char *buf, size_t size)
  *Attach the buffer from which data will be read in all subsequent operations.*
- void close ()
  *closes this input stream and releases any resources associated with the stream*
- unsigned int read ()
  *Reads the next bit from the input stream.The operation fails if eof() is true.*
- unsigned int read (unsigned int nbits)
  *Reads the specified number of bits from the input stream.*
- void read (unsigned char *destination, unsigned int nbits)
  *Reads the specified number of bits from the input stream into the destination buffer, assuming that the input is byte-aligned.*
- void skip (unsigned int nbits)
  *Skip the next n bits while reading from the current position.*
- bool eof () const
  *Informs about the read cursor position.*
- void reset ()
  *Reposition the read pointer at the beginning of the stream.*
- unsigned int align ()
  *Align the read pointer to the closest byte boundary.*
- size_t available () const
  *Returns the number of bits that can be read from this stream starting form the current position.*

- size_t consumed () const

  *Returns the number of bits that have been read so far.*
- const unsigned char * getPointer () const

  *Return the current read pointer, assuming it is byte-aligned.*
- size_t getSize () const

  *Get the size of the attached buffer.*
- void jumpTo (size_t position)

  *Jump to the indicated absolute position (in bits).*

### 7.2.1 Detailed Description

This class represents an input stream of bits.

**Author**

Massimo Balestri, Andrea Varesio, Marco Vecchietti

**Date**

2002

### 7.2.2 Constructor & Destructor Documentation

#### 7.2.2.1 mpeg7cdvs::BitInputStream::BitInputStream ( )

Create an empty object.

The open method must be first called to actually use the object.

#### 7.2.2.2 mpeg7cdvs::BitInputStream::BitInputStream ( const unsigned char * *buf*, size_t *size* )

Create and initialize a BitInputStream.

Attach the buffer from which data will be read in all subsequent operations.

**Parameters**

| | |
|---|---|
| *buf* | the buffer from which the data will be read |
| *size* | the size of the buffer in bytes (minimum size is 4 bytes) |

#### 7.2.2.3 virtual mpeg7cdvs::BitInputStream::~BitInputStream ( ) [virtual]

Close the stream, if not yet done, and destroy the object.

### 7.2.3 Member Function Documentation

#### 7.2.3.1 unsigned int mpeg7cdvs::BitInputStream::align ( )

Align the read pointer to the closest byte boundary.

If the read pointer is already aligned, the read pointer is not changed.

**Returns**

the number of skipped bits

#### 7.2.3.2 size_t mpeg7cdvs::BitInputStream::available ( ) const

Returns the number of bits that can be read from this stream starting form the current position.

**Returns**

the number of bits that can be read.

#### 7.2.3.3 void mpeg7cdvs::BitInputStream::close ( )

closes this input stream and releases any resources associated with the stream

#### 7.2.3.4 size_t mpeg7cdvs::BitInputStream::consumed ( ) const

Returns the number of bits that have been read so far.

**Returns**

the number of bits read so far.

#### 7.2.3.5 bool mpeg7cdvs::BitInputStream::eof ( ) const

Informs about the read cursor position.

**Returns**

true if the end of the input buffer has been reached.

#### 7.2.3.6 const unsigned char* mpeg7cdvs::BitInputStream::getPointer ( ) const

Return the current read pointer, assuming it is byte-aligned.

**Returns**

the read pointer

#### 7.2.3.7 size_t mpeg7cdvs::BitInputStream::getSize ( ) const

Get the size of the attached buffer.

**Returns**

the size in bytes.

#### 7.2.3.8 void mpeg7cdvs::BitInputStream::jumpTo ( size_t *position* )

Jump to the indicated absolute position (in bits).

**Parameters**

| | |
|---|---|
| *position* | the number of bits to skip from the start of the buffer. |

#### 7.2.3.9 void mpeg7cdvs::BitInputStream::open ( const unsigned char * *buf*, size_t *size* )

Attach the buffer from which data will be read in all subsequent operations.

**Parameters**

| | |
|---|---|
| *buf* | the buffer from which the data will be read |
| *size* | the size of the buffer in bytes (minimum size is 4 bytes) |

#### 7.2.3.10 unsigned int mpeg7cdvs::BitInputStream::read ( )

Reads the next bit from the input stream.The operation fails if eof() is true.

**Returns**

the next bit from the input stream.

#### 7.2.3.11 unsigned int mpeg7cdvs::BitInputStream::read ( unsigned int *nbits* )

Reads the specified number of bits from the input stream.

The operation fails if eof() is true.

**Parameters**

| | |
|---|---|
| *nbits* | the number of bits to read (in the range 1..32) |

**Returns**

the next n bits from the input stream.

#### 7.2.3.12 void mpeg7cdvs::BitInputStream::read ( unsigned char * *destination*, unsigned int *nbits* )

Reads the specified number of bits from the input stream into the destination buffer, assuming that the input is byte-aligned.

The operation fails if eof() is true.

**Parameters**

| | |
|---|---|
| *destination* | the destination buffer |
| *nbits* | the number of bits to read (8*n, assuming n>0) |

#### 7.2.3.13 void mpeg7cdvs::BitInputStream::reset ( )

Reposition the read pointer at the beginning of the stream.

#### 7.2.3.14 void mpeg7cdvs::BitInputStream::skip ( unsigned int *nbits* )

Skip the next n bits while reading from the current position.

If the end of the buffer is reached or even surpassed, eof() will return true.

**Parameters**

| | |
|---|---|
| *nbits* | the number of bits to skip (0..MAXINT) |

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

- BitInputStream.h

## 7.3 mpeg7cdvs::BitOutputStream Class Reference

This class represents an output stream of bits.

`#include <BitOutputStream.h>`

### Public Member Functions

- BitOutputStream ()
  *Creates an empty object.*
- BitOutputStream (unsigned char *buf, size_t size)
  *Create and initialize a BitOutputStream.*
- virtual ~BitOutputStream ()
  *Closes the stream, if not yet done, and destroys the object.*
- void open (unsigned char *buf, size_t size)
  *Attaches the object to a buffer of known size.*
- size_t close ()
  *Close this input stream.*
- void flush (unsigned int fill)
  *Align to the next byte boundary and flushes this output stream forcing any buffered output bits to be written in the destination buffer.*
- void write (unsigned int bit)
  *writes one bit into the output stream.*
- void write (unsigned int value, unsigned int nbits)
  *Writes the specified number of bits into the input stream.*
- void write (unsigned char *source, unsigned int nbits)
  *Writes the specified number of bits from the source buffer into the output stream, assuming that the output is byte-aligned.*
- void reset ()
  *Reposition the write pointer at the beginning of the stream.*
- void skip (unsigned int nbits)
  *Skip the next n bits while writing into the current position.*
- bool eof () const
  *Informs about the write cursor position.*
- void align (unsigned int fill)
  *Align the write pointer to the closest byte boundary.*
- size_t available () const
  *Returns the number of bits that can be written into this stream starting form the current position.*
- size_t produced () const
  *Returns the number of bits that have been written so far.*
- unsigned char * getPointer () const
  *returns the current write pointer, assuming it is byte-aligned.*
- size_t getSize () const
  *Get the size of the attached buffer.*
- void jumpTo (size_t position)
  *Jump to the indicated absolute position (in bits).*

### 7.3.1 Detailed Description

This class represents an output stream of bits.

**Author**

Massimo Balestri, Andrea Varesio, Marco Vecchietti

**Date**

2002

### 7.3.2 Constructor & Destructor Documentation

#### 7.3.2.1 mpeg7cdvs::BitOutputStream::BitOutputStream ( )

Creates an empty object.

The open method must be first called to actually use the object.

#### 7.3.2.2 mpeg7cdvs::BitOutputStream::BitOutputStream ( unsigned char * *buf*, size_t *size* )

Create and initialize a BitOutputStream.

#### 7.3.2.3 virtual mpeg7cdvs::BitOutputStream::~BitOutputStream ( ) [virtual]

Closes the stream, if not yet done, and destroys the object.

### 7.3.3 Member Function Documentation

#### 7.3.3.1 void mpeg7cdvs::BitOutputStream::align ( unsigned int *fill* )

Align the write pointer to the closest byte boundary.

If the write pointer is already aligned, the write pointer is not changed.

**Parameters**

| | |
|---|---|
| *fill* | the value to be used in order to fill the missing bits (must be 0 or 1). |

#### 7.3.3.2 size_t mpeg7cdvs::BitOutputStream::available ( ) const

Returns the number of bits that can be written into this stream starting form the current position.

**Returns**

the number of bits that can be written.

#### 7.3.3.3 size_t mpeg7cdvs::BitOutputStream::close ( )

Close this input stream.

This method flushes data and releases any resources associated with the stream.

**Returns**

the total number of produced bits.

#### 7.3.3.4 bool mpeg7cdvs::BitOutputStream::eof ( ) const

Informs about the write cursor position.

**Returns**

true if the end of the output buffer has been reached.

#### 7.3.3.5 void mpeg7cdvs::BitOutputStream::flush ( unsigned int *fill* )

Align to the next byte boundary and flushes this output stream forcing any buffered output bits to be written in the destination buffer.

**Parameters**

| | |
|---|---|
| *fill* | the value to be used in order to fill the missing bits (must be 0 or 1). |

#### 7.3.3.6 unsigned char* mpeg7cdvs::BitOutputStream::getPointer ( ) const

returns the current write pointer, assuming it is byte-aligned.

#### 7.3.3.7 size_t mpeg7cdvs::BitOutputStream::getSize ( ) const

Get the size of the attached buffer.

**Returns**

the size in bytes.

#### 7.3.3.8 void mpeg7cdvs::BitOutputStream::jumpTo ( size_t *position* )

Jump to the indicated absolute position (in bits).

The absolute position must be byte-aligned.

**Parameters**

| | |
|---|---|
| *position* | the number of bits to skip from the start of the buffer. |

#### 7.3.3.9 void mpeg7cdvs::BitOutputStream::open ( unsigned char * *buf*, size_t *size* )

Attaches the object to a buffer of known size.

This will be the destination in which data will be written in all subsequent operations.

**Parameters**

| | |
|---|---|
| *buf* | the buffer in which the data will be written |
| *size* | the size of the buffer in bytes |

#### 7.3.3.10 size_t mpeg7cdvs::BitOutputStream::produced ( ) const

Returns the number of bits that have been written so far.

**Returns**

the number of bits written so far.

#### 7.3.3.11 void mpeg7cdvs::BitOutputStream::reset ( )

Reposition the write pointer at the beginning of the stream.

#### 7.3.3.12 void mpeg7cdvs::BitOutputStream::skip ( unsigned int *nbits* )

Skip the next n bits while writing into the current position.

This operation first flushes any buffered bits, then jumps to the new location. If the end of the buffer is reached or even surpassed, eof() will return true.

**Parameters**

| | |
|---|---|
| *nbits* | the number of bits to skip (0..MAXINT) |

#### 7.3.3.13 void mpeg7cdvs::BitOutputStream::write ( unsigned int *bit* )

writes one bit into the output stream.

The operation fails if eof() is true.

**Parameters**

| | |
|---|---|
| *bit* | the bit to write (must be 0 or 1) |

#### 7.3.3.14 void mpeg7cdvs::BitOutputStream::write ( unsigned int *value,* unsigned int *nbits* )

Writes the specified number of bits into the input stream.

The operation fails if eof() is true.

**Parameters**

| | |
|---|---|
| *value* | the value to be written into stream. |
| *nbits* | the number of bits to write (in the range 1..32) |

#### 7.3.3.15 void mpeg7cdvs::BitOutputStream::write ( unsigned char * *source,* unsigned int *nbits* )

Writes the specified number of bits from the source buffer into the output stream, assuming that the output is byte-aligned.

The operation fails if eof() is true.

**Parameters**

| | |
|---|---|
| *source* | the source buffer (MUST be unsigned char, do not cast int or short arrays!) |
| *nbits* | the number of bits to be copied from the source into this output stream (8∗n, assuming n>0) |

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

- BitOutputStream.h

## 7.4 mpeg7cdvs::Buffer Class Reference

A container class for a byte array, intended to replace all malloc() and new() instructions in the main code.

`#include <Buffer.h>`

### Public Member Functions

- Buffer ()
- virtual ~Buffer ()
- Buffer (size_t size)

  *create a buffer of the given size*
- Buffer (unsigned char *data, size_t size)

  *copy the given array into this Buffer*
- Buffer (const Buffer &)

  *copy the given Buffer into this Buffer*
- Buffer & operator= (const Buffer &)

  *assign a Buffer to another*
- void swap (Buffer &x)

  *swap the content of two Buffer(s)*
- void fill (unsigned char value=0)

  *fill a Buffer with the given value*
- size_t size () const

  *return the current size of the Buffer*
- bool resize (size_t newsize)

  *change buffer size; content is lost if newsize if less than the current size*
- bool empty () const

  *return true if the Buffer is empty*
- void clear ()

  *clear the Buffer*
- bool assign (const unsigned char *data, size_t size)

  *assign the given data to Buffer*
- bool equals (Buffer &buffer)

  *compare if two Buffer(s) are equal (i.e. if they have the same size and contain the same data)*
- unsigned char * data ()

  *access to Buffer's data (writable)*
- const unsigned char * data () const

  *access to Buffer's data (read only)*
- void read (const char *fname)

  *read Buffer from a file*
- void write (const char *fname) const

  *write Buffer to file*
- int compare (const Buffer &other) const

  *Compare this buffer with another; return the number of different bytes.*
- bool operator== (const Buffer &other) const

  *compare if two Buffer(s) are equal (i.e. if they have the same size and contain the same data)*

### 7.4.1 Detailed Description

A container class for a byte array, intended to replace all malloc() and new() instructions in the main code.

This class properly deallocates memory when an exception is thrown.

**Author**

Massimo Balestri

**Date**

2013

### 7.4.2 Constructor & Destructor Documentation

#### 7.4.2.1 mpeg7cdvs::Buffer::Buffer ( )

#### 7.4.2.2 virtual mpeg7cdvs::Buffer::~Buffer ( ) [virtual]

#### 7.4.2.3 mpeg7cdvs::Buffer::Buffer ( size_t *size* )

create a buffer of the given size

#### 7.4.2.4 mpeg7cdvs::Buffer::Buffer ( unsigned char * *data*, size_t *size* )

copy the given array into this Buffer

#### 7.4.2.5 mpeg7cdvs::Buffer::Buffer ( const Buffer & )

copy the given Buffer into this Buffer

### 7.4.3 Member Function Documentation

#### 7.4.3.1 bool mpeg7cdvs::Buffer::assign ( const unsigned char * *data*, size_t *size* )

assign the given data to Buffer

#### 7.4.3.2 void mpeg7cdvs::Buffer::clear ( )

clear the Buffer

#### 7.4.3.3 int mpeg7cdvs::Buffer::compare ( const Buffer & *other* ) const

Compare this buffer with another; return the number of different bytes.

**Parameters**

| | |
|---|---|
| *other* | the other Buffer |

**Returns**

the number of differences; zero if no difference is found.

#### 7.4.3.4 unsigned char* mpeg7cdvs::Buffer::data ( )

access to Buffer's data (writable)

#### 7.4.3.5 const unsigned char* mpeg7cdvs::Buffer::data ( ) const

access to Buffer's data (read only)

#### 7.4.3.6 bool mpeg7cdvs::Buffer::empty ( ) const

return true if the Buffer is empty

#### 7.4.3.7 bool mpeg7cdvs::Buffer::equals ( Buffer & *buffer* )

compare if two Buffer(s) are equal (i.e. if they have the same size and contain the same data)

#### 7.4.3.8 void mpeg7cdvs::Buffer::fill ( unsigned char *value* = 0 )

fill a Buffer with the given value

#### 7.4.3.9 Buffer& mpeg7cdvs::Buffer::operator= ( const Buffer & )

assign a Buffer to another

#### 7.4.3.10 bool mpeg7cdvs::Buffer::operator== ( const Buffer & *other* ) const

compare if two Buffer(s) are equal (i.e. if they have the same size and contain the same data)

#### 7.4.3.11 void mpeg7cdvs::Buffer::read ( const char * *fname* )

read Buffer from a file

#### 7.4.3.12 bool mpeg7cdvs::Buffer::resize ( size_t *newsize* )

change buffer size; content is lost if newsize if less than the current size

#### 7.4.3.13 size_t mpeg7cdvs::Buffer::size ( ) const

return the current size of the Buffer

#### 7.4.3.14 void mpeg7cdvs::Buffer::swap ( Buffer & *x* )

swap the content of two Buffer(s)

#### 7.4.3.15 void mpeg7cdvs::Buffer::write ( const char * *fname* ) const

write Buffer to file

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

- Buffer.h

## 7.5 mpeg7cdvs::CdvsClient Class Reference

Interface to the client-side functionality of the CDVS Library.

`#include <CdvsInterface.h>`

### Public Member Functions

- virtual ~CdvsClient ()

- virtual unsigned int encode (CdvsDescriptor &output, int width, int height, const unsigned char *input) const =0

  *Encode the luminance component of an image producing a CDVS descriptor.*

### Static Public Member Functions

- static CdvsClient * cdvsClientFactory (const CdvsConfiguration *config, int mode)

  *Create an instance of a CDVS Client producing descriptors according to the indicated mode.*

## 7.5.1 Detailed Description

Interface to the client-side functionality of the CDVS Library.

**Author**

Massimo Balestri

**Date**

2014

## 7.5.2 Constructor & Destructor Documentation

### 7.5.2.1 virtual mpeg7cdvs::CdvsClient::~CdvsClient ( ) [inline], [virtual]

## 7.5.3 Member Function Documentation

### 7.5.3.1 static CdvsClient* mpeg7cdvs::CdvsClient::cdvsClientFactory ( const CdvsConfiguration * *config,* int *mode* ) [static]

Create an instance of a CDVS Client producing descriptors according to the indicated mode.

The calling entity takes ownership of the instance (i.e. must delete the instance when not used anymore).

**Parameters**

| | |
|---|---|
| *config* | the parameter configuration that will be used to produce descriptors. |
| *mode* | mode of the descriptors produced by the client instance. |

**Returns**

a pointer to the Cdvs Client instance

### 7.5.3.2 virtual unsigned int mpeg7cdvs::CdvsClient::encode ( CdvsDescriptor & *output,* int *width,* int *height,* const unsigned char * *input* ) const [pure virtual]

Encode the luminance component of an image producing a CDVS descriptor.

**Parameters**

| | |
|---|---|
| *output* | the output CDVS descriptor |

| | |
|---|---|
| *width* | width of the image |
| *height* | height of the image |
| *input* | the buffer containing the luminance component of the image (Y component, 8 bit per pixel) |

**Returns**

the actual size of the encoded CDVS descriptor

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

- CdvsInterface.h

## 7.6 mpeg7cdvs::CdvsConfiguration Class Reference

Interface to all configuration parameters for clients and servers.

`#include <CdvsInterface.h>`

### Public Member Functions

- virtual ~CdvsConfiguration ()
- virtual const Parameters & getParameters (int mode) const =0

  *Get one of the Parameters instances (note that this class keeps an instance of all parameters for all modes).*
- virtual Parameters & setParameters (int mode)=0

  *Set some Parameters value for a specific mode.*

### Static Public Member Functions

- static CdvsConfiguration * cdvsConfigurationFactory (const char *configfile=NULL)

  *Create an instance of a CDVS configuration containing all default coding/decoding parameters.*
- static int getMode (int descLen)

  *Get the mode ID corresponding to a specific descriptor length.*

### 7.6.1 Detailed Description

Interface to all configuration parameters for clients and servers.

**Author**

Massimo Balestri

**Date**

2014

### 7.6.2 Constructor & Destructor Documentation

#### 7.6.2.1 virtual mpeg7cdvs::CdvsConfiguration::~CdvsConfiguration ( ) [inline], [virtual]

### 7.6.3 Member Function Documentation

#### 7.6.3.1 static CdvsConfiguration* mpeg7cdvs::CdvsConfiguration::cdvsConfigurationFactory ( const char * *configfile* = NULL ) [static]

Create an instance of a CDVS configuration containing all default coding/decoding parameters.

The configuration instance can be used to initialize a client or a server CDVS instance. The configuration can be modified using the setParameters method. The calling entity takes ownership of the instance (i.e. must delete the instance when not used anymore).

**Parameters**

| | |
|---|---|
| *configfile* | a file containing some or all parameters replacing the default values. |

**Returns**

a CdvsConfiguration instance

#### 7.6.3.2 static int mpeg7cdvs::CdvsConfiguration::getMode ( int *descLen* ) [static]

Get the mode ID corresponding to a specific descriptor length.

The relation between length and mode ID is provided according to the MPEG CDVS specification:

- mode 1: 512 bytes
- mode 2: 1024 bytes
- mode 3: 2048 bytes
- mode 4: 4096 bytes
- mode 5: 8192 bytes
- mode 6: 16384 bytes

**Parameters**

| | |
|---|---|
| *descLen* | the descriptor length (in bytes) |

**Returns**

the corresponding mode

#### 7.6.3.3 virtual const Parameters& mpeg7cdvs::CdvsConfiguration::getParameters ( int *mode* ) const [pure virtual]

Get one of the Parameters instances (note that this class keeps an instance of all parameters for all modes).

**Parameters**

| | |
|---|---|
| *mode* | the mode for which parameters are requested. |

**Returns**

a read-only instance of the parameters.

#### 7.6.3.4 virtual Parameters& mpeg7cdvs::CdvsConfiguration::setParameters ( int *mode* ) [pure virtual]

Set some Parameters value for a specific mode.

**Parameters**

| | |
|---|---|
| *mode* | the mode for which parameters are requested. |

**Returns**

a modifiable instance of the parameters.

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

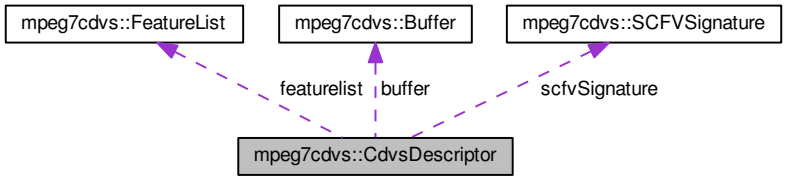
- CdvsInterface.h

## 7.7 mpeg7cdvs::CdvsDescriptor Class Reference

Helper class to read/write/check CDVS descriptors according to the syntax defined in ISO/IEC 15938-13.

`#include <CdvsDescriptor.h>`

Collaboration diagram for mpeg7cdvs::CdvsDescriptor:




### Public Member Functions

- CdvsDescriptor ()
- virtual ~CdvsDescriptor ()
- size_t encode (const Parameters &params, ImageBuffer &image, const SCFVFactory &g_factory)

  *Encode the CDVS descriptor.*
- size_t decode (const ParameterSet &pset)

  *Decode the CDVS descriptor.*
- int check () const

  *Check the conformance of the descriptor to the syntax defined in ISO/IEC 15938-13.*
- void clear ()

  *Clear all data.*
- void print (const char *title) const

  *Print the value of the syntax elements defined in ISO/IEC 15938-13.*
- unsigned int getVersionID () const

  *get the version ID*
- unsigned int getModeID () const

  *get the mode ID*
- bool getGlobalHasBitSelection () const

  *get the global descriptor bit selection flag*
- bool getGlobalHasVariance () const

  *get the global descriptor variance flag*

- bool getRelevanceBitsPresent () const

  *get the relevance bit flag*
- unsigned int getOriginalImageXResolution () const

  *get the original image X resolution*
- unsigned int getOriginalImageYResolution () const

  *get the original image Y resolution*
- unsigned int getNumberOfLocalDescriptors () const

  *get the number of local descriptors*
- unsigned int getHistogramCountSize () const

  *get the coordinate coding histogram count size*
- unsigned int getHistogramMapSizeX () const

  *get the coordinate coding map horizontal size*
- unsigned int getHistogramMapSizeY () const

  *get the coordinate coding map vertical size*
- void setVersionID (unsigned int vID)

  *set the version ID*
- void setModeID (unsigned int mID)

  *set the mode ID*
- void setGlobalHasBitSelection (bool gHasBS)

  *set the global descriptor bit selection flag*
- void setGlobalHasVariance (bool gHasV)

  *set the global descriptor variance flag*
- void setRelevanceBitsPresent (bool relevance)

  *set the relevance bit flag*
- void setOriginalImageXResolution (unsigned int oiXr)

  *set the original image X resolution*
- void setOriginalImageYResolution (unsigned int oiYr)

  *set the original image Y resolution*
- void setNumberOfLocalDescriptors (unsigned int nLD)

  *set the number of local descriptors*
- void setHistogramCountSize (unsigned int hCS)

  *set the coordinate coding histogram count size*
- void setHistogramMapSizeX (unsigned int hmsX)

  *set the coordinate coding map horizontal size*
- void setHistogramMapSizeY (unsigned int hmsY)

  *set the coordinate coding map vertical size*

## Data Fields

- Buffer buffer

  *the buffer containing the input/output bitstream*
- FeatureList featurelist

  *the list of key points*
- SCFVSignature scfvSignature

  *the global descriptor signature*

### 7.7.1 Detailed Description

Helper class to read/write/check CDVS descriptors according to the syntax defined in ISO/IEC 15938-13.

**Author**

Massimo Balestri (Telecom Italia)

**Date**

April, 2014

### 7.7.2 Constructor & Destructor Documentation

#### 7.7.2.1 mpeg7cdvs::CdvsDescriptor::CdvsDescriptor ( )

#### 7.7.2.2 virtual mpeg7cdvs::CdvsDescriptor::~CdvsDescriptor ( ) [virtual]

### 7.7.3 Member Function Documentation

#### 7.7.3.1 int mpeg7cdvs::CdvsDescriptor::check ( ) const

Check the conformance of the descriptor to the syntax defined in ISO/IEC 15938-13.

**Returns**

the number of out of range fields

#### 7.7.3.2 void mpeg7cdvs::CdvsDescriptor::clear ( )

Clear all data.

#### 7.7.3.3 size_t mpeg7cdvs::CdvsDescriptor::decode ( const ParameterSet & *pset* )

Decode the CDVS descriptor.

**Parameters**

| | |
|---|---|
| *pset* | set of parameters to apply for all modes from 0 to 6 |

**Returns**

the size of the consumed descriptor (bytes).

#### 7.7.3.4 size_t mpeg7cdvs::CdvsDescriptor::encode ( const Parameters & *params,* ImageBuffer & *image,* const SCFVFactory & *g_factory* )

Encode the CDVS descriptor.

This implementation does not extract all the key points from the image, but only those selected for transmission by the feature selection stage.

**Parameters**

| | |
|---|---|
| *params* | set of parameters to apply for one specific mode |
| *image* | the input image buffer |
| *g_factory* | the Global Descriptor factory instance that produces the GD signature for the specific mode selected in the parameters |

**Returns**

the size of the produced descriptor (bytes).

#### 7.7.3.5 bool mpeg7cdvs::CdvsDescriptor::getGlobalHasBitSelection ( ) const

get the global descriptor bit selection flag

#### 7.7.3.6 bool mpeg7cdvs::CdvsDescriptor::getGlobalHasVariance ( ) const

get the global descriptor variance flag

#### 7.7.3.7 unsigned int mpeg7cdvs::CdvsDescriptor::getHistogramCountSize ( ) const

get the coordinate coding histogram count size

#### 7.7.3.8 unsigned int mpeg7cdvs::CdvsDescriptor::getHistogramMapSizeX ( ) const

get the coordinate coding map horizontal size

#### 7.7.3.9 unsigned int mpeg7cdvs::CdvsDescriptor::getHistogramMapSizeY ( ) const

get the coordinate coding map vertical size

#### 7.7.3.10 unsigned int mpeg7cdvs::CdvsDescriptor::getModeID ( ) const

get the mode ID

#### 7.7.3.11 unsigned int mpeg7cdvs::CdvsDescriptor::getNumberOfLocalDescriptors ( ) const

get the number of local descriptors

#### 7.7.3.12 unsigned int mpeg7cdvs::CdvsDescriptor::getOriginalImageXResolution ( ) const

get the original image X resolution

#### 7.7.3.13 unsigned int mpeg7cdvs::CdvsDescriptor::getOriginalImageYResolution ( ) const

get the original image Y resolution

#### 7.7.3.14 bool mpeg7cdvs::CdvsDescriptor::getRelevanceBitsPresent ( ) const

get the relevance bit flag

#### 7.7.3.15 unsigned int mpeg7cdvs::CdvsDescriptor::getVersionID ( ) const

get the version ID

#### 7.7.3.16 void mpeg7cdvs::CdvsDescriptor::print ( const char * *title* ) const

Print the value of the syntax elements defined in ISO/IEC 15938-13.

**Parameters**

| | |
|---|---|
| *title* | the title to print as header information |

#### 7.7.3.17 void mpeg7cdvs::CdvsDescriptor::setGlobalHasBitSelection ( bool *gHasBS* )

set the global descriptor bit selection flag

#### 7.7.3.18 void mpeg7cdvs::CdvsDescriptor::setGlobalHasVariance ( bool *gHasV* )

set the global descriptor variance flag

#### 7.7.3.19 void mpeg7cdvs::CdvsDescriptor::setHistogramCountSize ( unsigned int *hCS* )

set the coordinate coding histogram count size

#### 7.7.3.20 void mpeg7cdvs::CdvsDescriptor::setHistogramMapSizeX ( unsigned int *hmsX* )

set the coordinate coding map horizontal size

#### 7.7.3.21 void mpeg7cdvs::CdvsDescriptor::setHistogramMapSizeY ( unsigned int *hmsY* )

set the coordinate coding map vertical size

#### 7.7.3.22 void mpeg7cdvs::CdvsDescriptor::setModeID ( unsigned int *mID* )

set the mode ID

#### 7.7.3.23 void mpeg7cdvs::CdvsDescriptor::setNumberOfLocalDescriptors ( unsigned int *nLD* )

set the number of local descriptors

#### 7.7.3.24 void mpeg7cdvs::CdvsDescriptor::setOriginalImageXResolution ( unsigned int *oiXr* )

set the original image X resolution

#### 7.7.3.25 void mpeg7cdvs::CdvsDescriptor::setOriginalImageYResolution ( unsigned int *oiYr* )

set the original image Y resolution

#### 7.7.3.26 void mpeg7cdvs::CdvsDescriptor::setRelevanceBitsPresent ( bool *relevance* )

set the relevance bit flag

#### 7.7.3.27 void mpeg7cdvs::CdvsDescriptor::setVersionID ( unsigned int *vID* )

set the version ID

### 7.7.4 Field Documentation

#### 7.7.4.1 Buffer mpeg7cdvs::CdvsDescriptor::buffer

the buffer containing the input/output bitstream

#### 7.7.4.2 FeatureList mpeg7cdvs::CdvsDescriptor::featurelist

the list of key points

#### 7.7.4.3 SCFVSignature mpeg7cdvs::CdvsDescriptor::scfvSignature

the global descriptor signature

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

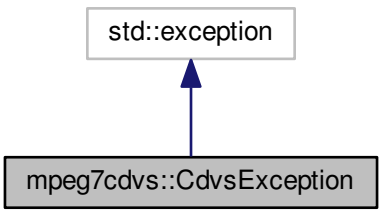
- CdvsDescriptor.h

## 7.8 mpeg7cdvs::CdvsException Class Reference

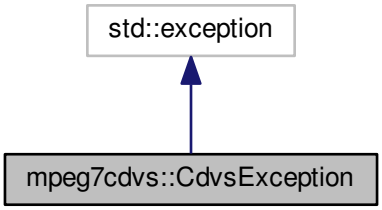
Class defining a specific exception for CDVS.

`#include <CdvsException.h>`

Inheritance diagram for mpeg7cdvs::CdvsException:



Collaboration diagram for mpeg7cdvs::CdvsException:




## Public Member Functions

- CdvsException (std::string str)

  *Create a new CDVS exception.*
- virtual ~CdvsException () throw ()
- const char * what () const throw ()

  *Get the exception message.*

### 7.8.1 Detailed Description

Class defining a specific exception for CDVS.

### 7.8.2 Constructor & Destructor Documentation

#### 7.8.2.1 mpeg7cdvs::CdvsException::CdvsException ( std::string *str* ) [inline]

Create a new CDVS exception.

**Parameters**

| | |
|---|---|
| *str* | the exception message string. |

#### 7.8.2.2 virtual mpeg7cdvs::CdvsException::~CdvsException ( ) throw ) [inline], [virtual]

### 7.8.3 Member Function Documentation

#### 7.8.3.1 const char* mpeg7cdvs::CdvsException::what ( ) const throw ) [inline]

Get the exception message.

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

- CdvsException.h

## 7.9 mpeg7cdvs::CDVSPOINT Struct Reference

A structure containing the x and y coordinate of a point in the image.

`#include <CdvsPoint.h>`

### Data Fields

- float x
  *the X coordinate*
- float y
  *the Y coordinate*

### 7.9.1 Detailed Description

A structure containing the x and y coordinate of a point in the image.

### 7.9.2 Field Documentation

#### 7.9.2.1 float mpeg7cdvs::CDVSPOINT::x

the X coordinate

#### 7.9.2.2 float mpeg7cdvs::CDVSPOINT::y

the Y coordinate

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

- CdvsPoint.h

## 7.10 mpeg7cdvs::CdvsServer Class Reference

Interface to the server-side functionality of the CDVS Library.

`#include <CdvsInterface.h>`

### Public Member Functions

- virtual ~CdvsServer ()
- virtual size_t decode (CdvsDescriptor &output, const char *fname) const =0
  *Decode a compressed query descriptor stored in a file.*
- virtual size_t decode (CdvsDescriptor &output, const unsigned char *bitstream=NULL, int size=0) const =0
  *Decode a compressed reference descriptor stored either in the bitstream parameter or in the CdvsDescriptor input/output Buffer.*
- virtual PointPairs match (const CdvsDescriptor &queryDescriptor, const CdvsDescriptor &refDescriptor, const CDVSPOINT *r_bbox=NULL, CDVSPOINT *proj_bbox=NULL, int matchType=MATCH_TYPE_DEFAULT) const =0
  *Pair-wise descriptor matching & localization function.*
- virtual PointPairs match (const CdvsDescriptor &queryDescriptor, unsigned int index, const CDVSPOINT *r_bbox=NULL, CDVSPOINT *proj_bbox=NULL, int matchType=MATCH_TYPE_LOCAL) const =0
  *Pair-wise descriptor matching & localization using a DB image as reference.*

- virtual void createDB (int mode, int reserve)=0

  *Create a Database of CDVS Descriptors for retrieval.*
- virtual unsigned int addDescriptorToDB (const CdvsDescriptor &refDescriptor, const char ∗referenceImageId)=0

  *Add the given reference descriptor to the Data Base of reference images.*
- virtual bool isDescriptorInDB (const char ∗referenceImageId) const =0

  *Verify if a given image is stored in the DB.*
- virtual bool replaceDescriptorInDB (const CdvsDescriptor &refDescriptor, const char ∗referenceImageId, const char ∗oldImageId=NULL)=0

  *Replace a given image in the DB with another one.*
- virtual void clearDB ()=0

  *Clear the DB removing all images.*
- virtual void commitDB ()=0

  *Commit all changes into the DB.*
- virtual void storeDB (const char ∗localname, const char ∗globalname) const =0

  *Store the Data Base permanently into a pair of files.*
- virtual void loadDB (const char ∗localname, const char ∗globalname)=0

  *Load the Data Base from a pair of files.*
- virtual size_t sizeofDB () const =0

  *Get the number of descriptors currently stored in the retrieval Data Base.*
- virtual int retrieve (std::vector< RetrievalData > &results, const CdvsDescriptor &queryDescriptor, unsigned int max_matches) const =0

  *Retrieval function.*
- virtual std::string getImageId (unsigned int index) const =0

  *Get the id corresponding to the given image index in the DB.*

### Static Public Member Functions

- static CdvsServer ∗ cdvsServerFactory (const CdvsConfiguration ∗config, bool twoWayMatch=true)

  *Create an instance of a CDVS Server for matching and retrieval of CDVS descriptors.*

## 7.10.1 Detailed Description

Interface to the server-side functionality of the CDVS Library.

**Author**

Massimo Balestri

**Date**

2014

## 7.10.2 Constructor & Destructor Documentation

### 7.10.2.1 virtual mpeg7cdvs::CdvsServer::~CdvsServer ( ) [inline], [virtual]

## 7.10.3 Member Function Documentation

### 7.10.3.1 virtual unsigned int mpeg7cdvs::CdvsServer::addDescriptorToDB ( const CdvsDescriptor & *refDescriptor,* const char ∗ *referenceImageId* ) [pure virtual]

Add the given reference descriptor to the Data Base of reference images.

**Parameters**

| | |
|---|---|
| *refDescriptor* | the reference descriptor |
| *referenceImage-Id* | the string that identifies this image; may be a pathname or a numeric ID but must be expressed as text. |

**Returns**

the index of the reference image in the DB

#### 7.10.3.2 static CdvsServer* mpeg7cdvs::CdvsServer::cdvsServerFactory ( const CdvsConfiguration * *config,* bool *twoWayMatch* = true ) [static]

Create an instance of a CDVS Server for matching and retrieval of CDVS descriptors.

The calling entity takes ownership of the instance (i.e. must delete the instance when not used anymore).

**Parameters**

| | |
|---|---|
| *config* | the configuration that will be used to produce descriptors. |
| *twoWayMatch* | select one-way or two-way matching; default is two-way. |

**Returns**

a pointer to the Cdvs Server instance

#### 7.10.3.3 virtual void mpeg7cdvs::CdvsServer::clearDB ( ) [pure virtual]

Clear the DB removing all images.

#### 7.10.3.4 virtual void mpeg7cdvs::CdvsServer::commitDB ( ) [pure virtual]

Commit all changes into the DB.

#### 7.10.3.5 virtual void mpeg7cdvs::CdvsServer::createDB ( int *mode,* int *reserve* ) [pure virtual]

Create a Database of CDVS Descriptors for retrieval.

**Parameters**

| | |
|---|---|
| *mode* | the mode identifier of all descriptors that will be stored in the DB; |
| *reserve* | the estimate number of CDVS Descriptor that will constitute the DB; the code will reserve a corresponding space in the DB. |

#### 7.10.3.6 virtual size_t mpeg7cdvs::CdvsServer::decode ( CdvsDescriptor & *output,* const char * *fname* ) const [pure virtual]

Decode a compressed query descriptor stored in a file.

**Parameters**

| | |
|---|---|
| *fname* | the input file name |
| *output* | the decoded CdvsDescriptor |

**Returns**

the size of the consumed descriptor (bytes).

#### 7.10.3.7 virtual size_t mpeg7cdvs::CdvsServer::decode ( CdvsDescriptor & *output*, const unsigned char * *bitstream* = NULL, int *size* = 0 ) const [pure virtual]

Decode a compressed reference descriptor stored either in the bitstream parameter or in the CdvsDescriptor input/output Buffer.

**Parameters**

| | |
|---|---|
| *output* | the decoded CdvsDescriptor |
| *bitstream* | a buffer containing an encoded CdvsDescriptor bitstream (optional parameter; if missing, the "buffer" member variable of CdvsDescriptor will be used instead) |
| *size* | size in bytes of the bitstream buffer (must be specified only if bitstream is not null) |

**Returns**

the size of the consumed descriptor (bytes).

#### 7.10.3.8 virtual std::string mpeg7cdvs::CdvsServer::getImageId ( unsigned int *index* ) const [pure virtual]

Get the id corresponding to the given image index in the DB.

**Parameters**

| | |
|---|---|
| *index* | the index in the DB of the image |

**Returns**

a string containing the identifier of the image

#### 7.10.3.9 virtual bool mpeg7cdvs::CdvsServer::isDescriptorInDB ( const char * *referenceImageId* ) const [pure virtual]

Verify if a given image is stored in the DB.

**Parameters**

| | |
|---|---|
| *referenceImageId* | the string that identifies this image; may be a pathname or a numeric ID but must be expressed as text. |

**Returns**

true if the image is present.

#### 7.10.3.10 virtual void mpeg7cdvs::CdvsServer::loadDB ( const char * *localname*, const char * *globalname* ) [pure virtual]

Load the Data Base from a pair of files.

**Parameters**

| | |
|---|---|
| *localname* | the name of the local descriptors file; |
| *globalname* | the name of the global descriptors file; |

#### 7.10.3.11 virtual PointPairs mpeg7cdvs::CdvsServer::match ( const CdvsDescriptor & *queryDescriptor,* const CdvsDescriptor & *refDescriptor,* const CDVSPOINT * *r_bbox* = NULL, CDVSPOINT * *proj_bbox* = NULL, int *matchType* = MATCH_TYPE_DEFAULT ) const [pure virtual]

Pair-wise descriptor matching & localization function.

**Parameters**

| | |
|---|---|
| *queryDescriptor* | the query descriptor |
| *refDescriptor* | the reference descriptor |
| *r_bbox* | bounding box of object of interest in the second (reference) image; replaced by the full image coordinates if NULL. |
| *proj_bbox* | buffer to contain parameters of bounding box for a match projected in the coordinate system of the first (query) image; ignored if NULL. |
| *matchType* | type of matching; may be MATCH_TYPE_DEFAULT, MATCH_TYPE_BOTH, MATCH_TYPE_LOCAL, MATCH_TYPE_GLOBAL. Default is MATCH_TYPE_DEFAULT in this case. |

**Returns**

an instance of PointPairs which contains all matching points, plus local and global scores.

#### 7.10.3.12 virtual PointPairs mpeg7cdvs::CdvsServer::match ( const CdvsDescriptor & *queryDescriptor,* unsigned int *index,* const CDVSPOINT * *r_bbox* = NULL, CDVSPOINT * *proj_bbox* = NULL, int *matchType* = MATCH_TYPE_LOCAL ) const [pure virtual]

Pair-wise descriptor matching & localization using a DB image as reference.

This method can be called after a retrieval operation, to localize the retrieved object(s) in the query image. The default match type in this case is MATCH_TYPE_LOCAL.

**Parameters**

| | |
|---|---|
| *queryDescriptor* | the query descriptor |
| *index* | index of the reference descriptor in the DB |
| *r_bbox* | bounding box of object of interest in the DB image; replaced by the full image coordinates if NULL. |
| *proj_bbox* | buffer to contain parameters of bounding box for a match projected in the coordinate system of the query image; ignored if NULL. |
| *matchType* | type of matching; may be MATCH_TYPE_DEFAULT, MATCH_TYPE_BOTH, MATCH_TYPE_LOCAL, MATCH_TYPE_GLOBAL. Default is MATCH_TYPE_LOCAL in this case. |

**Returns**

an instance of PointPairs which contains all matching points, plus local and global scores.

#### 7.10.3.13 virtual bool mpeg7cdvs::CdvsServer::replaceDescriptorInDB ( const CdvsDescriptor & *refDescriptor,* const char * *referenceImageId,* const char * *oldImageId* = NULL ) [pure virtual]

Replace a given image in the DB with another one.

If the image is not present no operation is performed.

**Parameters**

| | |
|---|---|
| *refDescriptor* | the reference descriptor of the new image |
| *referenceImage-Id* | the string that identifies the new image |
| *oldImageId* | the string that identifies the old image to be replaced; if NULL, referenceImageId will be also used as name of the image to replace |

**Returns**

true if the image was present (and its descriptor has been replaced).

#### 7.10.3.14 virtual int mpeg7cdvs::CdvsServer::retrieve ( std::vector< RetrievalData > & *results,* const CdvsDescriptor & *queryDescriptor,* unsigned int *max_matches* ) const [pure virtual]

Retrieval function.

Notes:

- it is assumed that database index is already pre-loaded and available through globals.
- query descriptor is also pre-loaded and passed via input parameters
- the task of this function is to produce a list of matching images in the database using only query descriptor, index, and descriptors of images stored in the database (included in the index)

  **Parameters**

  | | |
  |---|---|
  | *results* | vector of information data about matching images (in order of relevance) |
  | *queryDescriptor* | the query descriptor to be used as input query data of the retrieval operation |
  | *max_matches* | - maximum number of matches to include in the list of results |

  **Returns**

  number of matches found

#### 7.10.3.15 virtual size_t mpeg7cdvs::CdvsServer::sizeofDB ( ) const [pure virtual]

Get the number of descriptors currently stored in the retrieval Data Base.

**Returns**

the number of descriptors in the DB.

#### 7.10.3.16 virtual void mpeg7cdvs::CdvsServer::storeDB ( const char * *localname,* const char * *globalname* ) const [pure virtual]

Store the Data Base permanently into a pair of files.

**Parameters**

| | |
|---|---|
| *localname* | the name of the local descriptors file; |
| *globalname* | the name of the global descriptors file; |

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

- CdvsInterface.h

## 7.11 mpeg7cdvs::CompressedFeatureList Class Reference

Container class for all compressed features of an image.

`#include <FeatureList.h>`

### Public Member Functions

- CompressedFeatureList ()

  *default constructor*
- CompressedFeatureList (int nFeatures, int descLen)

  *parametric constructor (allocates memory)*
- CompressedFeatureList (const CompressedFeatureList &a)

  *copy constructor*
- virtual ~CompressedFeatureList ()
- CompressedFeatureList (const FeatureList &other, bool relevantOnly=false)

  *Copy constructor from FeatureList, optionally including relevance sorting.*
- CompressedFeatureList & operator= (CompressedFeatureList other)

  *Assignment operator.*
- void swap (CompressedFeatureList &other)

  *Swap this instance with another.*
- int nFeatures () const

  *Get the number of features.*
- int descrBytes () const

  *Get the size (number of bytes) of each feature stored in this compressed feature list.*
- void setFilename (const char *filename)

  *Set the name of the image (to be stored for subsequent retrieval).*
- int matchDescriptors_oneWay (PointPairs &pairs, const CompressedFeatureList &otherList, float ratioThreshold) const

  *Match the features of the current list with the ones contained in otherList in a one way fashion.*
- int matchDescriptors_twoWay (PointPairs &pairs, const CompressedFeatureList &otherList, float ratioThreshold) const

  *Match the features of the current list with the ones contained in otherList in a two way fashion.*
- std::streamoff readFromFile (char *filename)

  *Read an entire feature list from the given file.*
- std::streamoff read (std::istream &sin)

  *Read an entire feature list from the given input stream.*
- std::streamoff writeToFile (char *filename) const

  *Write an entire feature list into the given file.*
- std::streamoff write (std::ostream &sout) const

  *Write an entire feature list into the given output stream.*
- void print () const

  *Print a summary of the content.*

### Static Public Member Functions

- static int getDistance (const unsigned char *mine, const unsigned char *other, int nbytes)

  *Get the distance of one feature from another feature.*

### Data Fields

- unsigned short * Ycoord

  *the X coordinate of the ALP keypoint*
- unsigned short * Xcoord

  *the Y coordinate of the ALP keypoint*
- unsigned char * features

  *all compressed features*
- std::string imagefile

  *pathname of the image file.*
- int imageHeight

  *the (possibly scaled) image height.*
- int imageWidth

  *the (possibly scaled) image width.*
- int originalHeight

  *the original image height.*
- int originalWidth

  *the original image width.*

### Protected Attributes

- int numFeatures

  *number of features of this image*
- int nDescLength

  *descriptor length in bytes.*

## 7.11.1 Detailed Description

Container class for all compressed features of an image.

This is only used in the database in order to minimize the memory usage.

**Date**

2014

## 7.11.2 Constructor & Destructor Documentation

### 7.11.2.1 mpeg7cdvs::CompressedFeatureList::CompressedFeatureList ( )

default constructor

### 7.11.2.2 mpeg7cdvs::CompressedFeatureList::CompressedFeatureList ( int *nFeatures*, int *descLen* )

parametric constructor (allocates memory)

### 7.11.2.3 mpeg7cdvs::CompressedFeatureList::CompressedFeatureList ( const CompressedFeatureList & *a* )

copy constructor

#### 7.11.2.4 virtual mpeg7cdvs::CompressedFeatureList::~CompressedFeatureList ( ) [virtual]

#### 7.11.2.5 mpeg7cdvs::CompressedFeatureList::CompressedFeatureList ( const FeatureList & *other,* bool *relevantOnly* = false )

Copy constructor from FeatureList, optionally including relevance sorting.

**Parameters**

| | |
|---|---|
| *other* | the other FeatureList instance to copy |
| *relevantOnly* | if true, this method copies from a FeatureList instance only the features having the highest relevance |

### 7.11.3 Member Function Documentation

#### 7.11.3.1 int mpeg7cdvs::CompressedFeatureList::descrBytes ( ) const [inline]

Get the size (number of bytes) of each feature stored in this compressed feature list.

**Returns**

the size in bytes of the features.

References nDescLength.

#### 7.11.3.2 static int mpeg7cdvs::CompressedFeatureList::getDistance ( const unsigned char * *mine*, const unsigned char * *other*, int *nbytes* ) [static]

Get the distance of one feature from another feature.

**Parameters**

| | |
|---|---|
| *mine* | my feature |
| *other* | the other feature |
| *nbytes* | the number of bytes to use as input data |

**Returns**

the distance

#### 7.11.3.3 int mpeg7cdvs::CompressedFeatureList::matchDescriptors_oneWay ( PointPairs & *pairs*, const CompressedFeatureList & *otherList*, float *ratioThreshold* ) const

Match the features of the current list with the ones contained in otherList in a one way fashion.

The coordinates of the matching points are stored in the pairs parameter.

**Parameters**

| | |
|---|---|
| *pairs* | computed matching pairs of points |
| *otherList* | the other list. |
| *ratioThreshold* | the threshold used in the ratio test. |

**Returns**

the number of matched features.

#### 7.11.3.4 int mpeg7cdvs::CompressedFeatureList::matchDescriptors_twoWay ( PointPairs & *pairs*, const CompressedFeatureList & *otherList*, float *ratioThreshold* ) const

Match the features of the current list with the ones contained in otherList in a two way fashion.

The coordinates of the matching points are stored in the pairs parameter.

**Parameters**

| | |
|---|---|
| *pairs* | computed matching pairs of points |
| *otherList* | the other list. |
| *ratioThreshold* | the threshold used in the ratio test. |

**Returns**

the number of matched features.

#### 7.11.3.5 int mpeg7cdvs::CompressedFeatureList::nFeatures ( ) const [inline]

Get the number of features.

References numFeatures.

#### 7.11.3.6 CompressedFeatureList& mpeg7cdvs::CompressedFeatureList::operator= ( CompressedFeatureList *other* )

Assignment operator.

**Parameters**

| | |
|---|---|
| *other* | the other CompressedFeatureList instance |

**Returns**

a CompressedFeatureList instance

#### 7.11.3.7 void mpeg7cdvs::CompressedFeatureList::print ( ) const

Print a summary of the content.

#### 7.11.3.8 std::streamoff mpeg7cdvs::CompressedFeatureList::read ( std::istream & *sin* )

Read an entire feature list from the given input stream.

**Parameters**

| | |
|---|---|
| *sin* | the input stream. |

**Returns**

the number of bytes that have been read from the input stream.

#### 7.11.3.9 std::streamoff mpeg7cdvs::CompressedFeatureList::readFromFile ( char * *filename* )

Read an entire feature list from the given file.

**Parameters**

| | |
|---|---|
| *filename* | the pathname of the file containing the feature list. |

**Returns**

the number of bytes that have been read from the file.

#### 7.11.3.10 void mpeg7cdvs::CompressedFeatureList::setFilename ( const char * *filename* )

Set the name of the image (to be stored for subsequent retrieval).

**Parameters**

| | |
|---|---|
| *filename* | pathname of the image. |

#### 7.11.3.11 void mpeg7cdvs::CompressedFeatureList::swap ( CompressedFeatureList & *other* )

Swap this instance with another.

**Parameters**

| | |
|---|---|
| *other* | the other instance to swap with |

#### 7.11.3.12 std::streamoff mpeg7cdvs::CompressedFeatureList::write ( std::ostream & *sout* ) const

Write an entire feature list into the given output stream.

**Parameters**

| | |
|---|---|
| *sout* | the output stream. |

**Returns**

the number of bytes that have been written from the input stream.

#### 7.11.3.13 std::streamoff mpeg7cdvs::CompressedFeatureList::writeToFile ( char * *filename* ) const

Write an entire feature list into the given file.

**Parameters**

| | |
|---|---|
| *filename* | the pathname of the file where to store the feature list. |

**Returns**

the number of bytes that have been written into the file.

### 7.11.4 Field Documentation

#### 7.11.4.1 unsigned char* mpeg7cdvs::CompressedFeatureList::features

all compressed features

#### 7.11.4.2 std::string mpeg7cdvs::CompressedFeatureList::imagefile

pathname of the image file.

#### 7.11.4.3 int mpeg7cdvs::CompressedFeatureList::imageHeight

the (possibly scaled) image height.

#### 7.11.4.4 int mpeg7cdvs::CompressedFeatureList::imageWidth

the (possibly scaled) image width.

#### 7.11.4.5 int mpeg7cdvs::CompressedFeatureList::nDescLength [protected]

descriptor length in bytes.

Referenced by descrBytes().

#### 7.11.4.6 int mpeg7cdvs::CompressedFeatureList::numFeatures [protected]

number of features of this image

Referenced by nFeatures().

#### 7.11.4.7 int mpeg7cdvs::CompressedFeatureList::originalHeight

the original image height.

#### 7.11.4.8 int mpeg7cdvs::CompressedFeatureList::originalWidth

the original image width.

#### 7.11.4.9 unsigned short* mpeg7cdvs::CompressedFeatureList::Xcoord

the Y coordinate of the ALP keypoint

#### 7.11.4.10 unsigned short* mpeg7cdvs::CompressedFeatureList::Ycoord

the X coordinate of the ALP keypoint

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

- FeatureList.h

## 7.12 mpeg7cdvs::Feature Class Reference

Container class for the features of a single point (storing coordinates, scale, orientation, peak and descriptor of a point).

`#include <Feature.h>`

### Public Member Functions

- Feature (void)
- void toFile (FILE *file) const

  *Write the feature into a file.*
- void fromFile (FILE *file)

  *Write the feature into a file.*

### Data Fields

- float x

  *the X coordinate of the ALP keypoint*
- float y

*the Y coordinate of the ALP keypoint*

- float scale

  *the scale of the ALP keypoint*
- float orientation

  *the orientation of the ALP keypoint*
- float peak

  *the peak of the ALP keypoint*
- float curvRatio

  *the ratio of the curvatures*
- float curvSigma

  *the curvature at sigma*
- float descr [descrLength]

  *the SIFT descriptor of the ALP keypoint*
- float pdf

  *probability of this point to be matched*
- int spatialIndex

  *indicates the order of transmission of this point*
- unsigned short relevance

  *relevance of the keypoint, computed on the basis of his characteristics*
- int qdescr [descrLength]

  *the quantized (ternarized) descriptor values.*
- int octave

  *octave of this feature*
- int iscale

  *int scale*

### Static Public Attributes

- static const unsigned int descrLength = 128

  *the size of a feature (key point)*

## 7.12.1 Detailed Description

Container class for the features of a single point (storing coordinates, scale, orientation, peak and descriptor of a point).

**Author**

Gianluca Francini

**Date**

2011

## 7.12.2 Constructor & Destructor Documentation

### 7.12.2.1 mpeg7cdvs::Feature::Feature ( void )

## 7.12.3 Member Function Documentation

### 7.12.3.1 void mpeg7cdvs::Feature::fromFile ( FILE * *file* )

Write the feature into a file.

**Parameters**

| | |
|---|---|
| *file* | the output file. |

#### 7.12.3.2 void mpeg7cdvs::Feature::toFile ( FILE * *file* ) const

Write the feature into a file.

**Parameters**

| | |
|---|---|
| *file* | the output file. |

### 7.12.4 Field Documentation

#### 7.12.4.1 float mpeg7cdvs::Feature::curvRatio

the ratio of the curvatures

#### 7.12.4.2 float mpeg7cdvs::Feature::curvSigma

the curvature at sigma

#### 7.12.4.3 float mpeg7cdvs::Feature::descr[descrLength]

the SIFT descriptor of the ALP keypoint

#### 7.12.4.4 const unsigned int mpeg7cdvs::Feature::descrLength = 128 [static]

the size of a feature (key point)

#### 7.12.4.5 int mpeg7cdvs::Feature::iscale

int scale

#### 7.12.4.6 int mpeg7cdvs::Feature::octave

octave of this feature

#### 7.12.4.7 float mpeg7cdvs::Feature::orientation

the orientation of the ALP keypoint

#### 7.12.4.8 float mpeg7cdvs::Feature::pdf

probability of this point to be matched

#### 7.12.4.9 float mpeg7cdvs::Feature::peak

the peak of the ALP keypoint

#### 7.12.4.10 int mpeg7cdvs::Feature::qdescr[descrLength]

the quantized (ternarized) descriptor values.

#### 7.12.4.11 unsigned short mpeg7cdvs::Feature::relevance

relevance of the keypoint, computed on the basis of his characteristics

#### 7.12.4.12 float mpeg7cdvs::Feature::scale

the scale of the ALP keypoint

#### 7.12.4.13 int mpeg7cdvs::Feature::spatialIndex

indicates the order of transmission of this point

#### 7.12.4.14 float mpeg7cdvs::Feature::x

the X coordinate of the ALP keypoint

#### 7.12.4.15 float mpeg7cdvs::Feature::y

the Y coordinate of the ALP keypoint

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

- Feature.h

## 7.13 mpeg7cdvs::FeatureList Class Reference

Container class for all features of an image.

`#include <FeatureList.h>`

### Public Member Functions

- FeatureList ()
- void clear ()

  *Clear all memory.*
- void setResolution (int imgWidth, int imgHeight, int originalWidth, int originalHeight)

  *Store the resolution of the image from which the SIFT points were extracted.*
- int nFeatures () const

  *Get the number of features of the image.*
- void addFeature (const Feature &f)

  *Add a feature to the current list of features of the image.*
- void sortSpatialIndex ()

  *Sort the list of features on the basis of the spatial position of the points, used in the compression of coordinates.*
- void sortRelevance ()

  *Sort the list of features on the basis of the relevance.*
- int compareDescriptors (const FeatureList &otherList, bool compressed=false) const

*Compare the descriptor contained in this FeatureList with the one contained in otherList, and return the number of different values.*

- int compareCoordinates (const FeatureList &otherList, bool compressed=false, int blockWidth=1) const

  *Compare the coordinates contained in this FeatureList with the one contained in otherList, and return the number of different values.*
- int compareKeypoints (const FeatureList &otherList) const

  *Compare the key points properties contained in this FeatureList with the one contained in otherList, and return the number of different values.*
- void toFile (FILE *file) const

  *Write the entire FeatureList into a file.*
- void toFile (const char *filename) const

  *Write the entire FeatureList into a file.*
- void fromFile (FILE *file)

  *Read the entire FeatureList from a file.*
- void fromFile (const char *filename)

  *Read the entire FeatureList from a file.*
- void select (const std::vector< int > &indices)

  *Select a subset of features on the basis of the given indices; all other elements are discarded.*
- void selectFromTo (int startInd, int endInd)

  *Select a subset of features on the basis of the given range; all other elements are discarded.*
- void selectFirst (int n)

  *Select the first n features; all other elements are discarded.*
- void compress (int numberOfElementGroups)

  *Performs the compression of the SIFT descriptor.*
- void toBinary (BitOutputStream &writer, bool writeRelevance, int numFeatures)

  *Serialize FeatureList into a stream of bits.*
- void fromBinary (BitInputStream &reader, bool readRelevance)

  *De-serialize FeatureList from a stream of bits.*
- int computeMaxPoints (const Parameters &params, int targetBits)

  *Computes the maximum number of points to be added to the descriptor for a given bitrate.*
- void setRelevantPoints (int num)

  *Set the first n points as more relevant.*
- int getRelevantPoints () const

  *Get the number of relevant points.*
- void print () const

  *Print a summary of the featurelist content.*

## Data Fields

- unsigned int qdescr_size

  *The number of quantized elements in the key-point features (qdescr)*
- int imageHeight

  *the (possibly scaled) image height.*
- int imageWidth

  *the (possibly scaled) image width.*
- int originalHeight

  *the original image height.*
- int originalWidth

  *the original image width.*
- std::vector< Feature > features

  *the vector of features extracted from the image.*

### Static Public Attributes

- static const int MAX_NUM_FEATURES = 65536

  *theoretical limit set by the CDVS syntax*

### Friends

- class CompressedFeatureList

## 7.13.1 Detailed Description

Container class for all features of an image.

**Author**

Gianluca Francini

**Date**

2011

## 7.13.2 Constructor & Destructor Documentation

### 7.13.2.1 mpeg7cdvs::FeatureList::FeatureList ( )

## 7.13.3 Member Function Documentation

### 7.13.3.1 void mpeg7cdvs::FeatureList::addFeature ( const Feature & *f* )

Add a feature to the current list of features of the image.

**Parameters**

| | |
|---|---|
| *f* | the feature to be added. |

### 7.13.3.2 void mpeg7cdvs::FeatureList::clear ( )

Clear all memory.

### 7.13.3.3 int mpeg7cdvs::FeatureList::compareCoordinates ( const FeatureList & *otherList*, bool *compressed* = `false`, int *blockWidth* = `1` ) const

Compare the coordinates contained in this FeatureList with the one contained in otherList, and return the number of different values.

This is mainly used for debugging.

**Parameters**

| | |
|---|---|
| *otherList* | the other list to compare. |
| *compressed* | indicates if both descriptors are compressed |

| | |
|---|---|
| *blockWidth* | if compressed, indicates the quantization block width (in pixels) |

**Returns**

the number of different values in the two descriptors.

#### 7.13.3.4 int mpeg7cdvs::FeatureList::compareDescriptors ( const FeatureList & *otherList,* bool *compressed* = `false` ) const

Compare the descriptor contained in this FeatureList with the one contained in otherList, and return the number of different values.

This is mainly used for debugging.

**Parameters**

| | |
|---|---|
| *otherList* | the other list to compare. |
| *compressed* | indicates if both descriptors are compressed |

**Returns**

the number of different values in the two descriptors.

#### 7.13.3.5 int mpeg7cdvs::FeatureList::compareKeypoints ( const FeatureList & *otherList* ) const

Compare the key points properties contained in this FeatureList with the one contained in otherList, and return the number of different values.

This is mainly used for debugging.

**Parameters**

| | |
|---|---|
| *otherList* | the other list to compare. |

**Returns**

the number of different values in the two lists.

#### 7.13.3.6 void mpeg7cdvs::FeatureList::compress ( int *numberOfElementGroups* )

Performs the compression of the SIFT descriptor.

**Parameters**

| | |
|---|---|
| *numberOfElementGroups* | the number of element groups of this descriptor |

#### 7.13.3.7 int mpeg7cdvs::FeatureList::computeMaxPoints ( const Parameters & *params,* int *targetBits* )

Computes the maximum number of points to be added to the descriptor for a given bitrate.

This method does not assume any pre-computed statistics, just try to encode the features and discover how many bits are used.

**Parameters**

| | |
|---|---|
| *params* | the current running parameters |
| *targetBits* | the target number of bits to fill |

**Returns**

the number of points

#### 7.13.3.8 void mpeg7cdvs::FeatureList::fromBinary ( BitInputStream & *reader*, bool *readRelevance* )

De-serialize FeatureList from a stream of bits.

**Parameters**

| | |
|---|---|
| *reader* | the bitstream reader object. |
| *readRelevance* | read also the relevance value, used for the higher querylengths |

#### 7.13.3.9 void mpeg7cdvs::FeatureList::fromFile ( FILE * *file* )

Read the entire FeatureList from a file.

**Parameters**

| | |
|---|---|
| *file* | the input file. |

#### 7.13.3.10 void mpeg7cdvs::FeatureList::fromFile ( const char * *filename* )

Read the entire FeatureList from a file.

**Parameters**

| | |
|---|---|
| *filename* | the input filename. |

#### 7.13.3.11 int mpeg7cdvs::FeatureList::getRelevantPoints ( ) const

Get the number of relevant points.

**Returns**

the number of relevant points (generally smaller than the total number of key points).

#### 7.13.3.12 int mpeg7cdvs::FeatureList::nFeatures ( ) const

Get the number of features of the image.

**Returns**

the number of features currently stored in the features vector.

#### 7.13.3.13 void mpeg7cdvs::FeatureList::print ( ) const

Print a summary of the featurelist content.

#### 7.13.3.14 void mpeg7cdvs::FeatureList::select ( const std::vector< int > & *indices* )

Select a subset of features on the basis of the given indices; all other elements are discarded.

**Parameters**

| | |
|---|---|
| *indices* | indices of elements to keep. |

#### 7.13.3.15 void mpeg7cdvs::FeatureList::selectFirst ( int *n* )

Select the first n features; all other elements are discarded.

**Parameters**

| | |
|---|---|
| *n* | the number of elements to keep. |

#### 7.13.3.16 void mpeg7cdvs::FeatureList::selectFromTo ( int *startInd,* int *endInd* )

Select a subset of features on the basis of the given range; all other elements are discarded.

**Parameters**

| | |
|---|---|
| *startInd* | first elements to keep. |
| *endInd* | last elements to keep. |

#### 7.13.3.17 void mpeg7cdvs::FeatureList::setRelevantPoints ( int *num* )

Set the first n points as more relevant.

**Parameters**

| | |
|---|---|
| *num* | the number of releval points. |

#### 7.13.3.18 void mpeg7cdvs::FeatureList::setResolution ( int *imgWidth,* int *imgHeight,* int *originalWidth,* int *originalHeight* )

Store the resolution of the image from which the SIFT points were extracted.

**Parameters**

| | |
|---|---|
| *imgWidth* | the (possibly scaled) image width. |
| *imgHeight* | the (possibly scaled) image height. |
| *originalWidth* | the width of the original image. |
| *originalHeight* | the height the original image. |

#### 7.13.3.19 void mpeg7cdvs::FeatureList::sortRelevance ( )

Sort the list of features on the basis of the relevance.

#### 7.13.3.20 void mpeg7cdvs::FeatureList::sortSpatialIndex ( )

Sort the list of features on the basis of the spatial position of the points, used in the compression of coordinates.

#### 7.13.3.21 void mpeg7cdvs::FeatureList::toBinary ( BitOutputStream & *writer,* bool *writeRelevance,* int *numFeatures* )

Serialize FeatureList into a stream of bits.

**Parameters**

| | |
|---|---|
| *writer* | the bitstream writer object. |
| *writeRelevance* | write also the relevance value, used for the higher querylengths. |
| *numFeatures* | the number of features to encode. |

#### 7.13.3.22 void mpeg7cdvs::FeatureList::toFile ( FILE * *file* ) const

Write the entire FeatureList into a file.

**Parameters**

| | |
|---|---|
| *file* | the output file. |

#### 7.13.3.23 void mpeg7cdvs::FeatureList::toFile ( const char * *filename* ) const

Write the entire FeatureList into a file.

**Parameters**

| | |
|---|---|
| *filename* | the output filename. |

### 7.13.4 Friends And Related Function Documentation

#### 7.13.4.1 friend class CompressedFeatureList [friend]

### 7.13.5 Field Documentation

#### 7.13.5.1 std::vector<Feature> mpeg7cdvs::FeatureList::features

the vector of features extracted from the image.

#### 7.13.5.2 int mpeg7cdvs::FeatureList::imageHeight

the (possibly scaled) image height.

#### 7.13.5.3 int mpeg7cdvs::FeatureList::imageWidth

the (possibly scaled) image width.

#### 7.13.5.4 const int mpeg7cdvs::FeatureList::MAX_NUM_FEATURES = 65536 [static]

theoretical limit set by the CDVS syntax

#### 7.13.5.5 int mpeg7cdvs::FeatureList::originalHeight

the original image height.

#### 7.13.5.6 int mpeg7cdvs::FeatureList::originalWidth

the original image width.

#### 7.13.5.7 unsigned int mpeg7cdvs::FeatureList::qdescr_size

The number of quantized elements in the key-point features (qdescr)

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

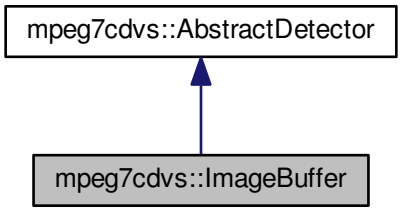
- FeatureList.h

## 7.14 mpeg7cdvs::ImageBuffer Class Reference

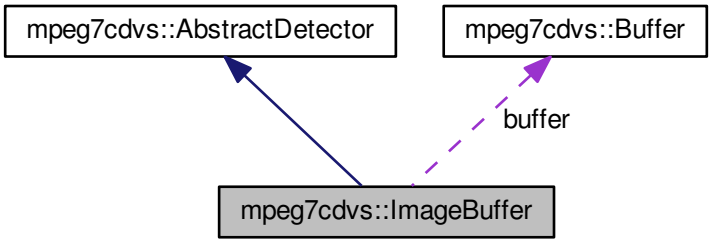
A container class for a bidimensional image; it's the base class of all keypoint detector classes.

`#include <ImageBuffer.h>`

Inheritance diagram for mpeg7cdvs::ImageBuffer:




Collaboration diagram for mpeg7cdvs::ImageBuffer:




### Public Member Functions

- ImageBuffer ()
- virtual ~ImageBuffer ()
- void swap (ImageBuffer &other)

  *swap the content of two ImageBuffer(s)*
- void read (int width, int height, const unsigned char *buffer)

*Read a planar luminance image from a buffer.*

- void resample (ImageBuffer &dest) const

  *Convert this image into a destination image having a different resolution by filtering and sampling the original image.*

- void resample (double rfactor)

  *Resample this image using the given reduction factor (the original image is discarded).*

- void resampleIfGreater (int maxSize)

  *Resample this image if either the horizontal or the vertical dimension of the image is greater that the given maximum size.*

### Static Public Member Functions

- static void print (const std::vector< Feature > &f, const char *source)

  *Print the given list of features.*

- static void printHeader (const char *source, size_t npoints)

  *Print the header of printDescr().*

- static void printDescr (const Feature &d)

  *Print the descriptor of the given feature.*

- static void writeBMP (const char *filename, const float *source, int w, int h)

  *write a BMP file containing the given image (luminance only).*

- static void writeRawData (const char *filename, const float *source, int w, int h)

  *write a raw data file containing h, w and the source float matrix.*

### Data Fields

- Buffer buffer

  *buffer containing the image data*

### Protected Member Functions

- bool resize (int newheight, int newwidth)

  *Resize the current image.*

### Static Protected Member Functions

- static unsigned int scalarQuantize (float value, const float *data, size_t size)

  *Perform scalar quantization on given data.*

- static float fastScalarQuantize (float value, const float *data, const float *output, size_t size)

  *Perform scalar quantization on given data and return the corresponding output.*

- static float fastInterpolate (float value, const float *data, const float *output, size_t size)

  *Perform scalar quantization then interpolate the output.*

- static bool sortPdfPredicate (const Feature &a, const Feature &b)

  *Predicate used to order features.*

- static bool sortPredicate (const float &a, const float &b)

  *Predicate used to order float values.*

### 7.14.1 Detailed Description

A container class for a bidimensional image; it's the base class of all keypoint detector classes.

This class properly deallocates memory when an exception is thrown.

**Author**

Giovanni Cordara, Massimo Balestri

**Date**

2013

### 7.14.2 Constructor & Destructor Documentation

#### 7.14.2.1 mpeg7cdvs::ImageBuffer::ImageBuffer ( )

#### 7.14.2.2 virtual mpeg7cdvs::ImageBuffer::~ImageBuffer ( ) [virtual]

### 7.14.3 Member Function Documentation

#### 7.14.3.1 static float mpeg7cdvs::ImageBuffer::fastInterpolate ( float *value,* const float * *data,* const float * *output,* size_t *size* ) [static], [protected]

Perform scalar quantization then interpolate the output.

**Parameters**

| | |
|---|---|
| *value* | the value to quantize |
| *data* | the quantization centroids |
| *output* | the corresponding output values (probabilities) |
| *size* | the size of the quantization centroid array |

**Returns**

the interpolated probability corresponding to the given value

#### 7.14.3.2 static float mpeg7cdvs::ImageBuffer::fastScalarQuantize ( float *value,* const float * *data,* const float * *output,* size_t *size* ) [static], [protected]

Perform scalar quantization on given data and return the corresponding output.

**Parameters**

| | |
|---|---|
| *value* | the value to quantize |
| *data* | the quantization centroids |
| *output* | the corresponding output values (probabilities) |
| *size* | the size of the quantization centroid array |

**Returns**

the probability corresponding to the given value

#### 7.14.3.3 static void mpeg7cdvs::ImageBuffer::print ( const std::vector< Feature > & *f,* const char * *source* ) [static]

Print the given list of features.

**Parameters**

| | |
|---|---|
| *f* | a vector of features |
| *source* | the name of the keypoint detector that produced the given list of features |

#### 7.14.3.4 static void mpeg7cdvs::ImageBuffer::printDescr ( const Feature & *d* ) [static]

Print the descriptor of the given feature.

**Parameters**

| | |
|---|---|
| *d* | the feature |

#### 7.14.3.5 static void mpeg7cdvs::ImageBuffer::printHeader ( const char * *source*, size_t *npoints* ) [static]

Print the header of printDescr().

**Parameters**

| | |
|---|---|
| *source* | the name of the detector under test |
| *npoints* | the number of points that will be printed |

#### 7.14.3.6 void mpeg7cdvs::ImageBuffer::read ( int *width*, int *height*, const unsigned char * *buffer* )

Read a planar luminance image from a buffer.

This method can be used only if the image is already available as an 8-bit planar luminance buffer.

**Parameters**

| | |
|---|---|
| *width* | width of the image |
| *height* | height of the image |
| *buffer* | the buffer containing the luminance component of the image |

#### 7.14.3.7 void mpeg7cdvs::ImageBuffer::resample ( ImageBuffer & *dest* ) const

Convert this image into a destination image having a different resolution by filtering and sampling the original image.

**Parameters**

| | |
|---|---|
| *dest* | the destination image |

**Exceptions**

| | |
|---|---|
| *CdvsException* | in case or error |

#### 7.14.3.8 void mpeg7cdvs::ImageBuffer::resample ( double *rfactor* )

Resample this image using the given reduction factor (the original image is discarded).

**Parameters**

| | |
|---|---|
| *rfactor* | the reduction factor (must be $< 1$) |

**Exceptions**

| | |
|---|---|
| *CdvsException* | in case or error |

#### 7.14.3.9 void mpeg7cdvs::ImageBuffer::resampleIfGreater ( int *maxSize* )

Resample this image if either the horizontal or the vertical dimension of the image is greater that the given maximum size.

**Parameters**

| | |
|---|---|
| *maxSize* | the maximum size to set |

**Exceptions**

| | |
|---|---|
| *CdvsException* | in case or error |

#### 7.14.3.10 bool mpeg7cdvs::ImageBuffer::resize ( int *newheight*, int *newwidth* ) [protected]

Resize the current image.

**Parameters**

| | |
|---|---|
| *newheight* | the new height |
| *newwidth* | the new width |

**Returns**

true if successful

#### 7.14.3.11 static unsigned int mpeg7cdvs::ImageBuffer::scalarQuantize ( float *value*, const float * *data*, size_t *size* ) [static], [protected]

Perform scalar quantization on given data.

**Parameters**

| | |
|---|---|
| *value* | the value to quantize |
| *data* | the quantization centroids |
| *size* | the size of the quantization centroid array |

**Returns**

the index such that the distance between value and data[index] is minimum

#### 7.14.3.12 static bool mpeg7cdvs::ImageBuffer::sortPdfPredicate ( const Feature & *a*, const Feature & *b* ) [static], [protected]

Predicate used to order features.

**Parameters**

| | |
|---|---|
| *a* | first element to compare |
| *b* | second element to compare |

**Returns**

true if a $>$ b

#### 7.14.3.13 static bool mpeg7cdvs::ImageBuffer::sortPredicate ( const float & *a*, const float & *b* ) [static], [protected]

Predicate used to order float values.

**Parameters**

| | |
|---|---|
| *a* | first element to compare |
| *b* | second element to compare |

**Returns**

true if a $>$ b

#### 7.14.3.14 void mpeg7cdvs::ImageBuffer::swap ( ImageBuffer & *other* )

swap the content of two ImageBuffer(s)

**Parameters**

| | |
|---|---|
| *other* | the other ImageBuffer instance |

#### 7.14.3.15 static void mpeg7cdvs::ImageBuffer::writeBMP ( const char * *filename*, const float * *source*, int *w*, int *h* ) [static]

write a BMP file containing the given image (luminance only).

#### 7.14.3.16 static void mpeg7cdvs::ImageBuffer::writeRawData ( const char * *filename*, const float * *source*, int *w*, int *h* ) [static]

write a raw data file containing h, w and the source float matrix.

### 7.14.4 Field Documentation

#### 7.14.4.1 Buffer mpeg7cdvs::ImageBuffer::buffer

buffer containing the image data

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

- ImageBuffer.h

## 7.15 mpeg7cdvs::LookUpTable Class Reference

A simple look up table implementation, to perform a bit count very quickly.

`#include <SCFVIndex.h>`

### Public Member Functions

- LookUpTable ()

### Data Fields

- char f [(1<< 16)]
  *the look up table*

#### 7.15.1 Detailed Description

A simple look up table implementation, to perform a bit count very quickly.

#### 7.15.2 Constructor & Destructor Documentation

##### 7.15.2.1 mpeg7cdvs::LookUpTable::LookUpTable ( )

#### 7.15.3 Field Documentation

##### 7.15.3.1 char mpeg7cdvs::LookUpTable::f[(1<< 16)]

the look up table

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

- SCFVIndex.h

## 7.16 mpeg7cdvs::Parameters Class Reference

Container for all encoding/decoding parameters associated to each target bitrate defined by MPEG CDVS.

`#include <Parameters.h>`

### Public Member Functions

- Parameters (void)
- ~Parameters (void)
- int readParameters (const char *filename, int mode)
  *Read text file and load parameters related to a specified mode.*
- int readParameters (int mode)
  *Load hard-coded parameters related to a specified mode.*
- unsigned int getModeID () const
  *Get the modeID of this set of parameters.*

### Static Public Member Functions

- static void readAll (const char *filename, Parameters params[])
  *Read all modes form a text file into the given vector of parameters.*
- static void readAll (Parameters params[])
  *Load all modes using default parameters into the given vector of parameters.*

## Data Fields

- int descLength

  *length in bytes of the CDVS descriptor (i.e. 512, 1024, 2048)*
- int resizeMaxSize

  *maximum size of one side of the image*
- int blockWidth

  *coordinate coding: spatial resolution of the coordinates (max error = blockWidth/2)*
- int ctxTableIdx

  *coordinate coding: index of the context table to use*
- char modeExt [40]

  *descriptor extension*
- unsigned int selectMaxPoints

  *feature extraction: max number of points used to describe an image*
- unsigned int numRelevantPoints

  *feature extraction: number of points considered relevant in the retrieval process*
- int numberOfElementGroups

  *feature compression: number of element groups in a compressed local feature descriptor*
- float ratioThreshold

  *DISTRAT: threshold for descriptor matching.*
- unsigned int minNumInliers

  *DISTRAT: min number of inliers after the geometric check.*
- double wmThreshold

  *Weighted matching threshold.*
- double wmThreshold2Way

  *Two way matching weighted threshold.*
- double wmMixed

  *Weighted matching threshold for mixed cases.*
- double wmMixed2Way

  *Two way weighted matching threshold for mixed cases.*
- int debugLevel

  *0 = off, 1 = on (quiet), 2 = on (verbose), 3 = verbose + dump files*
- int ransacNumTests

  *RANSAC: number of iterations in RANSAC.*
- float ransacThreshold

  *RANSAC: distortion threshold to be used by RANSAC.*
- unsigned int chiSquarePercentile

  *percentile used in DISTRAT for Chi-square computation*
- int retrievalLoops

  *number of loops performed in the final stage of the retrieval process*
- double wmRetrieval

  *Weighted matching threshold for retrieval.*
- double wmRetrieval2Way

  *Two way weighted matching threshold for retrieval.*
- int retrievalMaxPoints

  *max number of points used in the retrieval experiment*
- int queryExpansionLoops

  *number of query expansion loops to perform in the retrieval experiment*
- float scfvThreshold

  *threshold value to control the sparsity of scfv vector – add by linjie*
- bool hasVar

*indicates if using the gradient vector w.r.t the variance of Gaussian function – add by linjie*

- float locationBits

  *average bits per key point to encode location information;*
- bool hasBitSelection

  *indicates if the Global Descriptor uses the bit selection algorithm to reduce its size*
- float gdThreshold

  *global descriptor threshold*
- float gdThresholdMixed

  *global descriptor threshold for mixed cases*

### Static Public Attributes

- static const int nBits = 8

  *number of bits to represent the mode ID*
- static const int nModes = 7

  *the max number of processing modes*

## 7.16.1 Detailed Description

Container for all encoding/decoding parameters associated to each target bitrate defined by MPEG CDVS.

The actual value of each parameter is read from a text file, although default values are provided in this class. Each set of parameters is associated to a profile, and a single instance of this class may only contain parameters for a single profile. Please note that changing any parameter in the parameters file may break compatibility between encoder and decoder.

**Author**

Giovanni Cordara, Massimo Balestri

**Date**

2011

## 7.16.2 Constructor & Destructor Documentation

### 7.16.2.1 mpeg7cdvs::Parameters::Parameters ( void )

### 7.16.2.2 mpeg7cdvs::Parameters::~Parameters ( void )

## 7.16.3 Member Function Documentation

### 7.16.3.1 unsigned int mpeg7cdvs::Parameters::getModeID ( ) const

Get the modeID of this set of parameters.

The mode id cannot be changed; it is read from the parameters file.

**Returns**

the mode id value

### 7.16.3.2 static void mpeg7cdvs::Parameters::readAll ( const char * *filename,* Parameters *params[]* ) [static]

Read all modes form a text file into the given vector of parameters.

**Parameters**

| | |
|---|---|
| *filename* | filename pathname of the file containing the specific parameter values. If NULL it is ignored. |
| *params* | the vector of parameters to fill. |

#### 7.16.3.3 static void mpeg7cdvs::Parameters::readAll ( Parameters *params[]* ) [static]

Load all modes using default parameters into the given vector of parameters.

**Parameters**

| | |
|---|---|
| *params* | the vector of parameters to fill. |

#### 7.16.3.4 int mpeg7cdvs::Parameters::readParameters ( const char * *filename,* int *mode* )

Read text file and load parameters related to a specified mode.

A maximum of nModes are supported.

**Parameters**

| | |
|---|---|
| *filename* | pathname of the file containing the specific parameter values. If NULL it is ignored. |
| *mode* | one of the MPEG CDVS supported modes, from 0 to 6. |

**Returns**

0 if successful, an error code otherwise.

#### 7.16.3.5 int mpeg7cdvs::Parameters::readParameters ( int *mode* )

Load hard-coded parameters related to a specified mode.

A maximum of nModes are supported.

**Parameters**

| | |
|---|---|
| *mode* | one of the MPEG CDVS supported modes, from 0 to 6. |

**Returns**

0 if successful, an error code otherwise.

### 7.16.4 Field Documentation

#### 7.16.4.1 int mpeg7cdvs::Parameters::blockWidth

coordinate coding: spatial resolution of the coordinates (max error = blockWidth/2)

#### 7.16.4.2 unsigned int mpeg7cdvs::Parameters::chiSquarePercentile

percentile used in DISTRAT for Chi-square computation

#### 7.16.4.3 int mpeg7cdvs::Parameters::ctxTableIdx

coordinate coding: index of the context table to use

#### 7.16.4.4 int mpeg7cdvs::Parameters::debugLevel

0 = off, 1 = on (quiet), 2 = on (verbose), 3 = verbose + dump files

#### 7.16.4.5 int mpeg7cdvs::Parameters::descLength

length in bytes of the CDVS descriptor (i.e. 512, 1024, 2048)

#### 7.16.4.6 float mpeg7cdvs::Parameters::gdThreshold

global descriptor threshold

#### 7.16.4.7 float mpeg7cdvs::Parameters::gdThresholdMixed

global descriptor threshold for mixed cases

#### 7.16.4.8 bool mpeg7cdvs::Parameters::hasBitSelection

indicates if the Global Descriptor uses the bit selection algorithm to reduce its size

#### 7.16.4.9 bool mpeg7cdvs::Parameters::hasVar

indicates if using the gradient vector w.r.t the variance of Gaussian function – add by linjie

#### 7.16.4.10 float mpeg7cdvs::Parameters::locationBits

average bits per key point to encode location information;

#### 7.16.4.11 unsigned int mpeg7cdvs::Parameters::minNumInliers

DISTRAT: min number of inliers after the geometric check.

#### 7.16.4.12 char mpeg7cdvs::Parameters::modeExt[40]

descriptor extension

#### 7.16.4.13 const int mpeg7cdvs::Parameters::nBits = 8 [static]

number of bits to represent the mode ID

#### 7.16.4.14 const int mpeg7cdvs::Parameters::nModes = 7 [static]

the max number of processing modes

#### 7.16.4.15 int mpeg7cdvs::Parameters::numberOfElementGroups

feature compression: number of element groups in a compressed local feature descriptor

#### 7.16.4.16 unsigned int mpeg7cdvs::Parameters::numRelevantPoints

feature extraction: number of points considered relevant in the retrieval process

#### 7.16.4.17 int mpeg7cdvs::Parameters::queryExpansionLoops

number of query expansion loops to perform in the retrieval experiment

#### 7.16.4.18 int mpeg7cdvs::Parameters::ransacNumTests

RANSAC: number of iterations in RANSAC.

#### 7.16.4.19 float mpeg7cdvs::Parameters::ransacThreshold

RANSAC: distortion threshold to be used by RANSAC.

#### 7.16.4.20 float mpeg7cdvs::Parameters::ratioThreshold

DISTRAT: threshold for descriptor matching.

#### 7.16.4.21 int mpeg7cdvs::Parameters::resizeMaxSize

maximum size of one side of the image

#### 7.16.4.22 int mpeg7cdvs::Parameters::retrievalLoops

number of loops performed in the final stage of the retrieval process

#### 7.16.4.23 int mpeg7cdvs::Parameters::retrievalMaxPoints

max number of points used in the retrieval experiment

#### 7.16.4.24 float mpeg7cdvs::Parameters::scfvThreshold

threshold value to control the sparsity of scfv vector – add by linjie

#### 7.16.4.25 unsigned int mpeg7cdvs::Parameters::selectMaxPoints

feature extraction: max number of points used to describe an image

#### 7.16.4.26 double mpeg7cdvs::Parameters::wmMixed

Weighted matching threshold for mixed cases.

#### 7.16.4.27 double mpeg7cdvs::Parameters::wmMixed2Way

Two way weighted matching threshold for mixed cases.

#### 7.16.4.28 double mpeg7cdvs::Parameters::wmRetrieval

Weighted matching threshold for retrieval.

#### 7.16.4.29 double mpeg7cdvs::Parameters::wmRetrieval2Way

Two way weighted matching threshold for retrieval.

#### 7.16.4.30 double mpeg7cdvs::Parameters::wmThreshold

Weighted matching threshold.

#### 7.16.4.31 double mpeg7cdvs::Parameters::wmThreshold2Way

Two way matching weighted threshold.

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

- Parameters.h

## 7.17 mpeg7cdvs::PointPairs Class Reference

Parameter class, used to pass around matched point coordinates.

`#include <PointPairs.h>`

### Public Member Functions

- PointPairs ()

  *default constructor*
- PointPairs (int maxPairs)

  *alternate constructor*
- PointPairs (const PointPairs &other)

  *copy constructor*
- PointPairs & operator= (PointPairs other)

  *assignment operator*
- virtual ~PointPairs ()

  *destructor*
- bool hasLocalizationInliers () const

  *Return true if the PointPairs instance contains localization information.*
- void addPair (float x_1, float x_2, float y_1, float y_2, double weight=0, int mtype=match_2way_DISJOINT1)

  *Add a new pair to the matched pairs and increment the nMatched count.*
- void toFullResolution (int query_maxres, int query_fullres, int ref_maxres, int ref_fullres)

  *Convert matched coordinates to the full resolution of the original image.*
- double getTotalWeight () const

  *Get the total weight of all matched points.*
- double getInlierWeight () const

  *Get the weight of inlier points (these are a subset of all points, which passed the geometric verification test).*

### Data Fields

- double local_score
  *matching score provided by local descriptors*
- double global_score
  *matching score provided by global descriptor*
- double local_threshold
  *local threshold used for weighted matching of keypoints*
- double global_threshold
  *global threshold used for matching the global descriptor*
- double score
  *final normalized score (in a range from 0 to 1)*
- int nMatched
  *actual number of matched points*
- int size
  *size of all buffers*
- float * x1
  *x-coordinates of matching points of the first image*
- float * x2
  *y-coordinates of matching points of the first image*
- float * y1
  *x-coordinates of matching points of the second image*
- float * y2
  *y-coordinates of matching points of the second image*
- double * weights
  *weights of each match*
- int * match_dirs
  *indicates the direction of matching (A<=>B, A=>B, B=>A) in 2-way matching (not used in 1-way matching)*
- int nInliers
  *indicates the number of pairs which actually match according to the geometric verification*
- int * inlierIndexes
  *indicates the indices of the pairs that have passed the geometric verification*

## 7.17.1 Detailed Description

Parameter class, used to pass around matched point coordinates.

Used by the matching methods in the Feature*List* classes.

**Author**

Emanuele Plebani

**Date**

2012

## 7.17.2 Constructor & Destructor Documentation

### 7.17.2.1 mpeg7cdvs::PointPairs::PointPairs ( )

default constructor

#### 7.17.2.2 mpeg7cdvs::PointPairs::PointPairs ( int *maxPairs* )

alternate constructor

#### 7.17.2.3 mpeg7cdvs::PointPairs::PointPairs ( const PointPairs & *other* )

copy constructor

#### 7.17.2.4 virtual mpeg7cdvs::PointPairs::~PointPairs ( ) [virtual]

destructor

### 7.17.3 Member Function Documentation

#### 7.17.3.1 void mpeg7cdvs::PointPairs::addPair ( float *x_1*, float *x_2*, float *y_1*, float *y_2*, double *weight* = 0, int *mtype* = match_2way_DISJOINT1 )

Add a new pair to the matched pairs and increment the nMatched count.

**Parameters**

| | |
|---|---|
| *x_1* | x-coordinate of matching point of the first image |
| *x_2* | y-coordinate of matching point of the first image |
| *y_1* | x-coordinate of matching point of the second image |
| *y_2* | y-coordinate of matching point of the second image |
| *weight* | weight of matching (defaults is zero if not used) |
| *mtype* | direction of matching for 2-way matching (A<=>B, A=>B or B=>A (defaults is A=>B if not used) |

#### 7.17.3.2 double mpeg7cdvs::PointPairs::getInlierWeight ( ) const

Get the weight of inlier points (these are a subset of all points, which passed the geometric verification test).

**Returns**

the weight of inlier points.

#### 7.17.3.3 double mpeg7cdvs::PointPairs::getTotalWeight ( ) const

Get the total weight of all matched points.

**Returns**

the total weight of all matched points.

#### 7.17.3.4 bool mpeg7cdvs::PointPairs::hasLocalizationInliers ( ) const

Return true if the PointPairs instance contains localization information.

#### 7.17.3.5 PointPairs& mpeg7cdvs::PointPairs::operator= ( PointPairs *other* )

assignment operator

#### 7.17.3.6 void mpeg7cdvs::PointPairs::toFullResolution ( int *query_maxres,* int *query_fullres,* int *ref_maxres,* int *ref_fullres* )

Convert matched coordinates to the full resolution of the original image.

This method may change the scale of the coordinates of matching points stored in x1, x2, y1, y2.

**Parameters**

| | |
|---|---|
| *query_maxres* | the greater dimension of the (possibly scaled) query image. |
| *query_fullres* | the greater dimension of the original query image. |
| *ref_maxres* | the greater dimension of the (possibly scaled) reference image. |
| *ref_fullres* | the greater dimension of the original reference image. |

### 7.17.4 Field Documentation

#### 7.17.4.1 double mpeg7cdvs::PointPairs::global_score

matching score provided by global descriptor

#### 7.17.4.2 double mpeg7cdvs::PointPairs::global_threshold

global threshold used for matching the global descriptor

#### 7.17.4.3 int* mpeg7cdvs::PointPairs::inlierIndexes

indicates the indices of the pairs that have passed the geometric verification

#### 7.17.4.4 double mpeg7cdvs::PointPairs::local_score

matching score provided by local descriptors

#### 7.17.4.5 double mpeg7cdvs::PointPairs::local_threshold

local threshold used for weighted matching of keypoints

#### 7.17.4.6 int* mpeg7cdvs::PointPairs::match_dirs

indicates the direction of matching (A<=>B, A=>B, B=>A) in 2-way matching (not used in 1-way matching)

#### 7.17.4.7 int mpeg7cdvs::PointPairs::nInliers

indicates the number of pairs which actually match according to the geometric verification

#### 7.17.4.8 int mpeg7cdvs::PointPairs::nMatched

actual number of matched points

#### 7.17.4.9 double mpeg7cdvs::PointPairs::score

final normalized score (in a range from 0 to 1)

#### 7.17.4.10 int mpeg7cdvs::PointPairs::size

size of all buffers

#### 7.17.4.11 double* mpeg7cdvs::PointPairs::weights

weights of each match

#### 7.17.4.12 float* mpeg7cdvs::PointPairs::x1

x-coordinates of matching points of the first image

#### 7.17.4.13 float* mpeg7cdvs::PointPairs::x2

y-coordinates of matching points of the first image

#### 7.17.4.14 float* mpeg7cdvs::PointPairs::y1

x-coordinates of matching points of the second image

#### 7.17.4.15 float* mpeg7cdvs::PointPairs::y2

y-coordinates of matching points of the second image

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

- PointPairs.h

## 7.18 mpeg7cdvs::RetrievalData Struct Reference

A structure containing the output of a retrieval operation.

`#include <CdvsPoint.h>`

### Data Fields

- unsigned int nMatched
  *number of matched points*
- unsigned int nInliers
  *number of inliers points*
- unsigned int index
  *index of this image in the image DB*
- float gScore
  *score assigned by the global descriptor matching*
- float fScore
  *score assigned by the local descriptors matching*

### 7.18.1 Detailed Description

A structure containing the output of a retrieval operation.

### 7.18.2 Field Documentation

#### 7.18.2.1 float mpeg7cdvs::RetrievalData::fScore

score assigned by the local descriptors matching

#### 7.18.2.2 float mpeg7cdvs::RetrievalData::gScore

score assigned by the global descriptor matching

#### 7.18.2.3 unsigned int mpeg7cdvs::RetrievalData::index

index of this image in the image DB

#### 7.18.2.4 unsigned int mpeg7cdvs::RetrievalData::nInliers

number of inliers points

#### 7.18.2.5 unsigned int mpeg7cdvs::RetrievalData::nMatched

number of matched points

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

- CdvsPoint.h

## 7.19 mpeg7cdvs::SCFVFactory Class Reference

A class to produce SCFV signatures.

`#include <SCFVIndex.h>`

### Public Member Functions

- SCFVFactory ()
- void init (const Parameters &params)

  *Initialize the class with the correct set of parameters.*
- void generateSCFV (const FeatureList &featureList, SCFVSignature &signature, int nNumFeatures) const

  *Generate a global descriptor signature using the given feature list.*
- bool hasVariance () const

  *Indicates if using the variance information of the Gaussian function.*
- bool hasBitSelection () const

  *Indicates if using the bit selection information.*

### 7.19.1 Detailed Description

A class to produce SCFV signatures.

### 7.19.2 Constructor & Destructor Documentation

#### 7.19.2.1 mpeg7cdvs::SCFVFactory::SCFVFactory ( )

### 7.19.3 Member Function Documentation

#### 7.19.3.1 void mpeg7cdvs::SCFVFactory::generateSCFV ( const FeatureList & *featureList,* SCFVSignature & *signature,* int *nNumFeatures* ) const

Generate a global descriptor signature using the given feature list.

**Parameters**

| | |
|---|---|
| *featureList* | the key points to use as input information |
| *signature* | the output signature |
| *nNumFeatures* | the number of features to encode |

#### 7.19.3.2 bool mpeg7cdvs::SCFVFactory::hasBitSelection ( ) const [inline]

Indicates if using the bit selection information.

**Returns**

true if using the bit selection information

#### 7.19.3.3 bool mpeg7cdvs::SCFVFactory::hasVariance ( ) const [inline]

Indicates if using the variance information of the Gaussian function.

**Returns**

true if using the variance information

#### 7.19.3.4 void mpeg7cdvs::SCFVFactory::init ( const Parameters & *params* )

Initialize the class with the correct set of parameters.

**Parameters**

| | |
|---|---|
| *params* | the input parameters to use |

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

- SCFVIndex.h

## 7.20 mpeg7cdvs::SCFVIndex Class Reference

A class to manage an indexed list of SCFV signatures.

`#include <SCFVIndex.h>`

### Public Member Functions

- SCFVIndex ()

- void append (const SCFVSignature &scfvSignature)

  *append the given SCFV signature to the current index*
- void replace (size_t index, const SCFVSignature &scfvSignature)

  *replace the given SCFV signature with the given one at the given index*
- void write (std::string sIndexName) const

  *write the SCFV index to file*
- void read (std::string sIndexName)

  *read the SCFV index from file*
- void query (const SCFVSignature &querySignature, std::vector< std::pair< double, unsigned int > > &vImageScoresNumbers, size_t numRankedOuput) const

  *Use a binary SCFV signature as a query to retrieve a ranked list of signatures matching the given one.*
- void query_bitselection (const SCFVSignature &querySignature, std::vector< std::pair< double, unsigned int > > &vImageScoresNumbers, size_t numRankedOuput) const

  *Use a subset of a binary SCFV signature as a query to retrieve a ranked list of signatures matching the given one.*
- void generateWeight (const SCFVSignature &querySignature, float *W2_log, float *W2_log_var, float weight_base) const

  *Produces an optional table of weights to reduce the importance of features that are too common.*
- size_t numberImages () const

  *Get the number of images (actually signatures) contained in this index.*
- const SCFVSignature & getImage (unsigned int index) const

  *Get the SCFV signature of a specific image.*
- void resize (size_t num)

  *Resize the index to num elements.*
- void reserve (size_t num)

  *Reserve memory for the given number of signatures.*
- void clear ()

  *Clear all signatures.*
- float matchImages (const SCFVSignature &signature1, const SCFVSignature &signature2, unsigned int *pNumWords1, unsigned int *pNumWords2, unsigned int *overlap) const

  *Match two signatures and return a matching score.*
- float matchImages_bitselection (const SCFVSignature &signature1, const SCFVSignature &signature2, unsigned int *pNumWords1, unsigned int *pNumWords2, unsigned int *overlap) const

  *Match two signatures applying bit selection and return a matching score.*
- void loadHammingWeight ()

  *Initialize the index with Hamming distance weights.*

### 7.20.1 Detailed Description

A class to manage an indexed list of SCFV signatures.

Includes methods to read/write/append SCFV signatures, to use a signature as a query, and to match two signatures.

### 7.20.2 Constructor & Destructor Documentation

#### 7.20.2.1 mpeg7cdvs::SCFVIndex::SCFVIndex ( )

### 7.20.3 Member Function Documentation

#### 7.20.3.1 void mpeg7cdvs::SCFVIndex::append ( const SCFVSignature & *scfvSignature* )

append the given SCFV signature to the current index

#### 7.20.3.2 void mpeg7cdvs::SCFVIndex::clear ( ) [inline]

Clear all signatures.

#### 7.20.3.3 void mpeg7cdvs::SCFVIndex::generateWeight ( const SCFVSignature & *querySignature,* float * *W2_log,* float * *W2_log_var,* float *weight_base* ) const

Produces an optional table of weights to reduce the importance of features that are too common.

**Parameters**

| | |
|---|---|
| *querySignature* | the query image signature |
| *W2_log* | (output) the logarithmic weight for mean values |
| *W2_log_var* | (output) the logarithmic weight for variance values |
| *weight_base* | the basic weight from which the table is produced. |

#### 7.20.3.4 const SCFVSignature& mpeg7cdvs::SCFVIndex::getImage ( unsigned int *index* ) const [inline]

Get the SCFV signature of a specific image.

**Parameters**

| | |
|---|---|
| *index* | index of the image in the database of images. |

**Returns**

the image signature

#### 7.20.3.5 void mpeg7cdvs::SCFVIndex::loadHammingWeight ( )

Initialize the index with Hamming distance weights.

#### 7.20.3.6 float mpeg7cdvs::SCFVIndex::matchImages ( const SCFVSignature & *signature1,* const SCFVSignature & *signature2,* unsigned int * *pNumWords1,* unsigned int * *pNumWords2,* unsigned int * *overlap* ) const

Match two signatures and return a matching score.

**Parameters**

| | |
|---|---|
| *signature1* | the first SCFV signature |
| *signature2* | the second SCFV signature |
| *pNumWords1* | the visited number of words of signature1 |
| *pNumWords2* | the visited number of words of signature2 |
| *overlap* | unused |

**Returns**

the matching score

#### 7.20.3.7 float mpeg7cdvs::SCFVIndex::matchImages_bitselection ( const SCFVSignature & *signature1,* const SCFVSignature & *signature2,* unsigned int * *pNumWords1,* unsigned int * *pNumWords2,* unsigned int * *overlap* ) const

Match two signatures applying bit selection and return a matching score.

**Parameters**

| | |
|---|---|
| *signature1* | the first SCFV signature |
| *signature2* | the second SCFV signature |
| *pNumWords1* | the visited number of words of signature1 |
| *pNumWords2* | the visited number of words of signature2 |
| *overlap* | unused |

**Returns**

the matching score

#### 7.20.3.8 size_t mpeg7cdvs::SCFVIndex::numberImages ( ) const [inline]

Get the number of images (actually signatures) contained in this index.

**Returns**

the number of images.

#### 7.20.3.9 void mpeg7cdvs::SCFVIndex::query ( const SCFVSignature & *querySignature,* std::vector< std::pair< double, unsigned int > > & *vImageScoresNumbers,* size_t *numRankedOuput* ) const

Use a binary SCFV signature as a query to retrieve a ranked list of signatures matching the given one.

**Parameters**

| | |
|---|---|
| *querySignature* | the query signature |
| *vImageScores-Numbers* | the output ordered list of images matching the query |
| *numRanked-Ouput* | the number of maximum output images required |

#### 7.20.3.10 void mpeg7cdvs::SCFVIndex::query_bitselection ( const SCFVSignature & *querySignature,* std::vector< std::pair< double, unsigned int > > & *vImageScoresNumbers,* size_t *numRankedOuput* ) const

Use a subset of a binary SCFV signature as a query to retrieve a ranked list of signatures matching the given one.

**Parameters**

| | |
|---|---|
| *querySignature* | the query signature |
| *vImageScores-Numbers* | the output ordered list of images matching the query |
| *numRanked-Ouput* | the number of maximum output images required |

#### 7.20.3.11 void mpeg7cdvs::SCFVIndex::read ( std::string *sIndexName* )

read the SCFV index from file

#### 7.20.3.12 void mpeg7cdvs::SCFVIndex::replace ( size_t *index,* const SCFVSignature & *scfvSignature* )

replace the given SCFV signature with the given one at the given index

#### 7.20.3.13 void mpeg7cdvs::SCFVIndex::reserve ( size_t *num* ) [inline]

Reserve memory for the given number of signatures.

**Parameters**

| | |
|---|---|
| *num* | the number of signatures to be reserved in the index. |

#### 7.20.3.14 void mpeg7cdvs::SCFVIndex::resize ( size_t *num* ) [inline]

Resize the index to num elements.

**Parameters**

| | |
|---|---|
| *num* | the number of elements required to be in the index |

#### 7.20.3.15 void mpeg7cdvs::SCFVIndex::write ( std::string *sIndexName* ) const

write the SCFV index to file

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

- SCFVIndex.h

## 7.21 mpeg7cdvs::SCFVSignature Class Reference

Container class for a Scalable Fisher Vector binary signature; allows reading/writing from/to a bitstream, fetching/storing from/into a file, and comparing a signature with another.

`#include <SCFVIndex.h>`

### Public Member Functions

- SCFVSignature (bool hasVar, bool hasBitSelection)
  *Constructor declaring if this signature contains variance information and bit selection.*
- void clear ()
  *clear all data*
- size_t size () const
  *get the size of the binary signature (uncompressed)*
- int compressedNumBits () const
  *get the number of bits of the encoded signature (compressed)*
- void write (BitOutputStream &out) const
  *write the binary signature into the given output stream*
- void read (BitInputStream &in)
  *read the binary signature from the given input stream*
- unsigned int getVisited () const
  *get the number of visited words*
- void setVisited ()
  *compute and store the correct number of visited words*
- float getNorm () const
  *get the norm of this signature*
- void setNorm ()
  *compute and store the correct norm for this signature*
- bool hasVar () const
  *tell if this signature has variance information*
- void hasVar (bool value)

*set this signature as one containing variance information (if value is true)*

- bool hasBitSelection () const

  *tell if this signature performs bit selection*

- void hasBitSelection (bool value)

  *set this signature as one performing bit selection (if value is true)*

- int compare (const SCFVSignature &other) const

  *compare two signatures (only for debugging)*

- void toFile (FILE *file) const

  *write the signature to file*

- void fromFile (FILE *file)

  *read the signature from file*

- void print () const

  *print a summary of the signature data*

### Data Fields

- unsigned int m_vWordBlock [numberCentroids]

  *Scalable Fisher Vector binary signature.*

- unsigned int m_vWordVarBlock [numberCentroids]

  *Scalable Fisher Vector binary variance.*

### Static Public Attributes

- static const unsigned int table_bit_selection [ ]

  *subset of bits used in low bitrate applications*

## 7.21.1 Detailed Description

Container class for a Scalable Fisher Vector binary signature; allows reading/writing from/to a bitstream, fetching/storing from/into a file, and comparing a signature with another.

## 7.21.2 Constructor & Destructor Documentation

### 7.21.2.1 mpeg7cdvs::SCFVSignature::SCFVSignature ( bool *hasVar,* bool *hasBitSelection* )

Constructor declaring if this signature contains variance information and bit selection.

**Parameters**

| | |
|---|---|
| *hasVar* | true if this signature contains variance information (used normally at high bitrates) |
| *hasBitSelection* | true if this signature performs bit selection (used normally at very low bitrates) |

## 7.21.3 Member Function Documentation

### 7.21.3.1 void mpeg7cdvs::SCFVSignature::clear ( )

clear all data

### 7.21.3.2 int mpeg7cdvs::SCFVSignature::compare ( const SCFVSignature & *other* ) const

compare two signatures (only for debugging)

#### 7.21.3.3 int mpeg7cdvs::SCFVSignature::compressedNumBits ( ) const

get the number of bits of the encoded signature (compressed)

#### 7.21.3.4 void mpeg7cdvs::SCFVSignature::fromFile ( FILE * *file* )

read the signature from file

#### 7.21.3.5 float mpeg7cdvs::SCFVSignature::getNorm ( ) const

get the norm of this signature

#### 7.21.3.6 unsigned int mpeg7cdvs::SCFVSignature::getVisited ( ) const

get the number of visited words

#### 7.21.3.7 bool mpeg7cdvs::SCFVSignature::hasBitSelection ( ) const

tell if this signature performs bit selection

#### 7.21.3.8 void mpeg7cdvs::SCFVSignature::hasBitSelection ( bool *value* )

set this signature as one performing bit selection (if value is true)

#### 7.21.3.9 bool mpeg7cdvs::SCFVSignature::hasVar ( ) const

tell if this signature has variance information

#### 7.21.3.10 void mpeg7cdvs::SCFVSignature::hasVar ( bool *value* )

set this signature as one containing variance information (if value is true)

#### 7.21.3.11 void mpeg7cdvs::SCFVSignature::print ( ) const

print a summary of the signature data

#### 7.21.3.12 void mpeg7cdvs::SCFVSignature::read ( BitInputStream & *in* )

read the binary signature from the given input stream

#### 7.21.3.13 void mpeg7cdvs::SCFVSignature::setNorm ( )

compute and store the correct norm for this signature

#### 7.21.3.14 void mpeg7cdvs::SCFVSignature::setVisited ( )

compute and store the correct number of visited words

#### 7.21.3.15 size_t mpeg7cdvs::SCFVSignature::size ( ) const

get the size of the binary signature (uncompressed)

#### 7.21.3.16 void mpeg7cdvs::SCFVSignature::toFile ( FILE * *file* ) const

write the signature to file

#### 7.21.3.17 void mpeg7cdvs::SCFVSignature::write ( BitOutputStream & *out* ) const

write the binary signature into the given output stream

### 7.21.4 Field Documentation

#### 7.21.4.1 unsigned int mpeg7cdvs::SCFVSignature::m_vWordBlock[numberCentroids]

Scalable Fisher Vector binary signature.

#### 7.21.4.2 unsigned int mpeg7cdvs::SCFVSignature::m_vWordVarBlock[numberCentroids]

Scalable Fisher Vector binary variance.

#### 7.21.4.3 const unsigned int mpeg7cdvs::SCFVSignature::table_bit_selection[] [static]

subset of bits used in low bitrate applications

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

- SCFVIndex.h

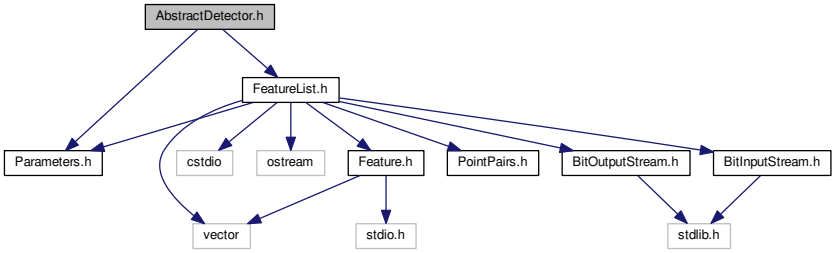
# Chapter 8

# File Documentation

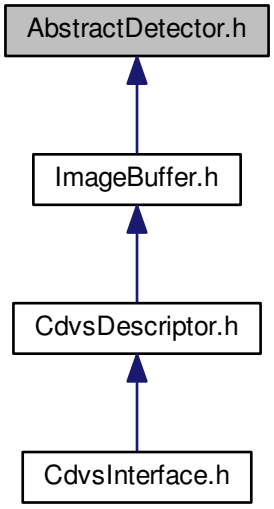
## 8.1 AbstractDetector.h File Reference

```
#include "FeatureList.h"
#include "Parameters.h"
```

Include dependency graph for AbstractDetector.h:



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




### Data Structures

- class mpeg7cdvs::AbstractDetector

  *Base class for keypoint detectors.*

### Namespaces

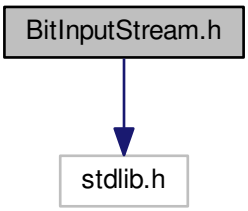
- mpeg7cdvs

  *Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

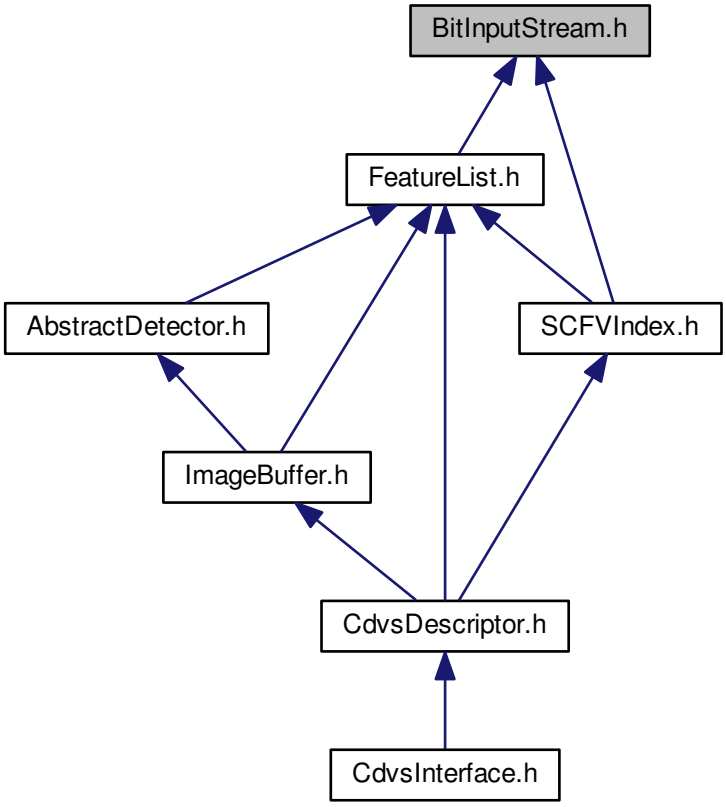
## 8.2 BitInputStream.h File Reference

```
#include <stdlib.h>
```

Include dependency graph for BitInputStream.h:




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




## Data Structures

- class mpeg7cdvs::BitInputStream

  *This class represents an input stream of bits.*

### Namespaces

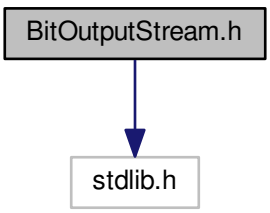
- mpeg7cdvs

*Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

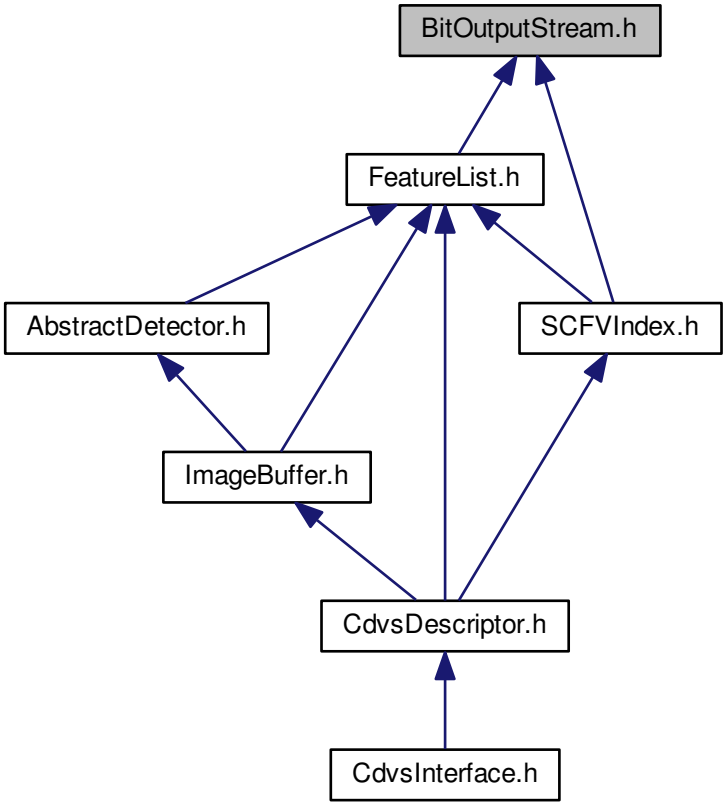
## 8.3 BitOutputStream.h File Reference

```
#include <stdlib.h>
```

Include dependency graph for BitOutputStream.h:



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




### Data Structures

- class mpeg7cdvs::BitOutputStream

  *This class represents an output stream of bits.*

### Namespaces

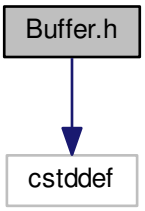
- mpeg7cdvs

  *Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

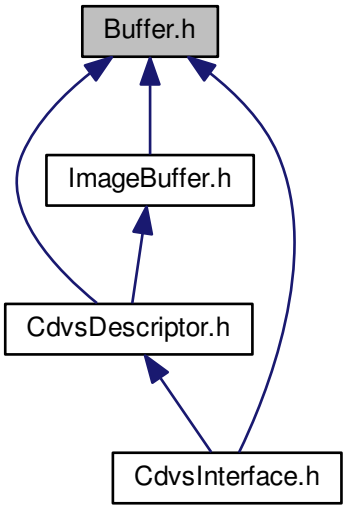
## 8.4 Buffer.h File Reference

```
#include <cstddef>
```

Include dependency graph for Buffer.h:




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




## Data Structures

- class mpeg7cdvs::Buffer

  *A container class for a byte array, intended to replace all malloc() and new() instructions in the main code.*

## Namespaces

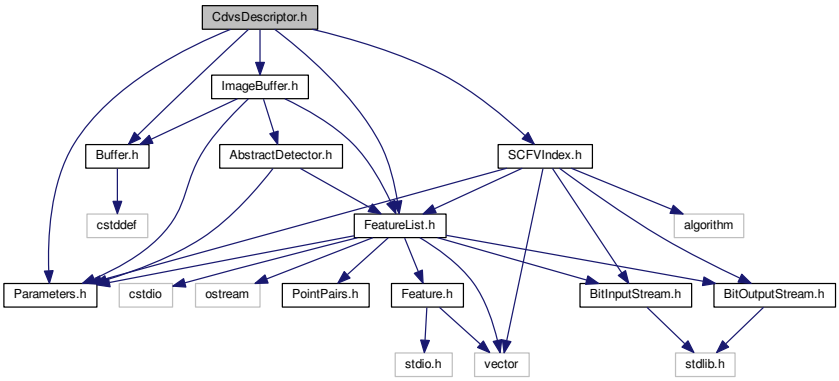
- mpeg7cdvs

  *Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

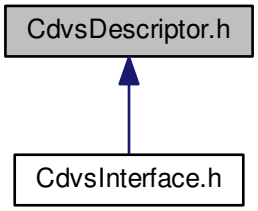
## 8.5 CdvsDescriptor.h File Reference

```
#include "Parameters.h"
#include "Buffer.h"
#include "ImageBuffer.h"
#include "FeatureList.h"
#include "SCFVIndex.h"
```

Include dependency graph for CdvsDescriptor.h:




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




### Data Structures

- class mpeg7cdvs::CdvsDescriptor

  *Helper class to read/write/check CDVS descriptors according to the syntax defined in ISO/IEC 15938-13.*

### Namespaces

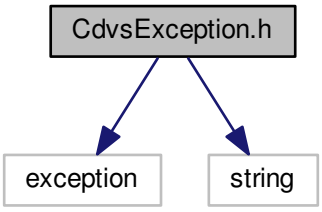
- mpeg7cdvs

  *Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

## 8.6 CdvsException.h File Reference

```
#include <exception>
#include <string>
```

Include dependency graph for CdvsException.h:




### Data Structures

- class mpeg7cdvs::CdvsException

  *Class defining a specific exception for CDVS.*

### Namespaces

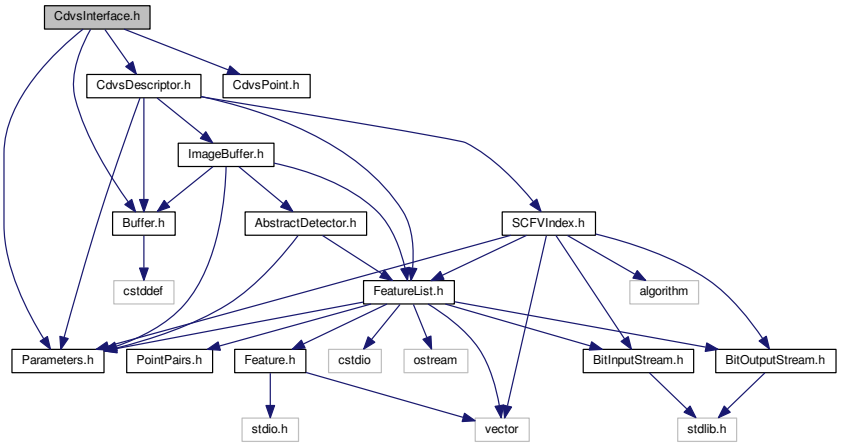
- mpeg7cdvs

  *Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

## 8.7 CdvsInterface.h File Reference

```
#include "Parameters.h"
#include "CdvsDescriptor.h"
#include "CdvsPoint.h"
#include "Buffer.h"
```

Include dependency graph for CdvsInterface.h:




### Data Structures

- class mpeg7cdvs::CdvsConfiguration

  *Interface to all configuration parameters for clients and servers.*
- class mpeg7cdvs::CdvsClient

  *Interface to the client-side functionality of the CDVS Library.*
- class mpeg7cdvs::CdvsServer

  *Interface to the server-side functionality of the CDVS Library.*

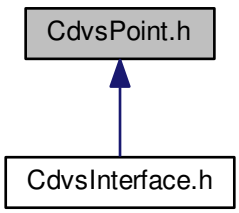
### Namespaces

- mpeg7cdvs

  *Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

## 8.8 CdvsPoint.h File Reference

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



### Data Structures

- struct mpeg7cdvs::CDVSPOINT

  *A structure containing the x and y coordinate of a point in the image.*

- struct mpeg7cdvs::RetrievalData

  *A structure containing the output of a retrieval operation.*

### Namespaces

- mpeg7cdvs

  *Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

### Enumerations

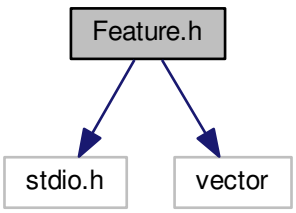
- enum { mpeg7cdvs::MATCH_TYPE_DEFAULT = 0, mpeg7cdvs::MATCH_TYPE_BOTH = 1, mpeg7cdvs::MATCH_TYPE_LOCAL = 2, mpeg7cdvs::MATCH_TYPE_GLOBAL = 3 }

  *Type of matching.*

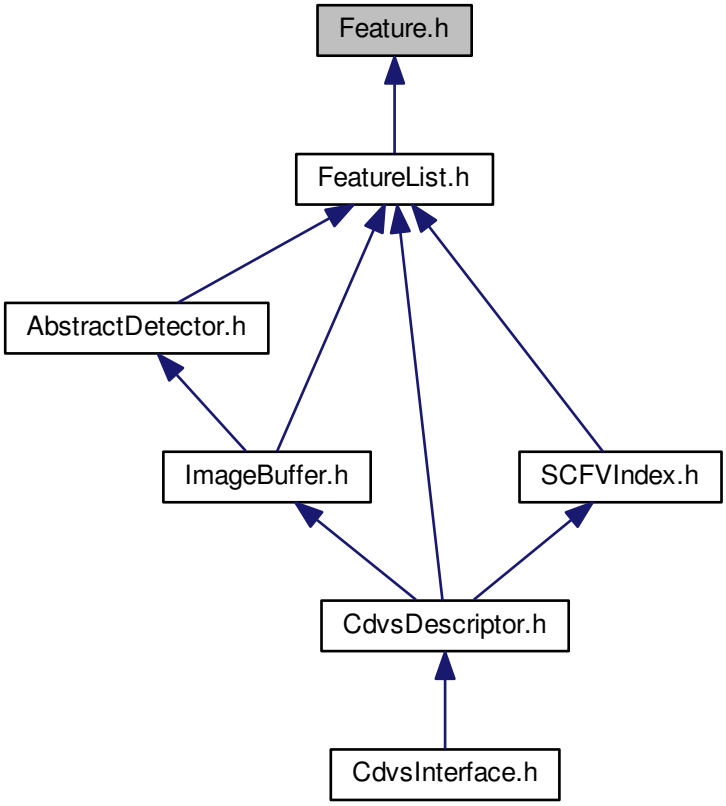
## 8.9 Feature.h File Reference

```
#include <stdio.h>
#include <vector>
```

Include dependency graph for Feature.h:



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




### Data Structures

- class mpeg7cdvs::Feature

  *Container class for the features of a single point (storing coordinates, scale, orientation, peak and descriptor of a point).*

### Namespaces

- mpeg7cdvs

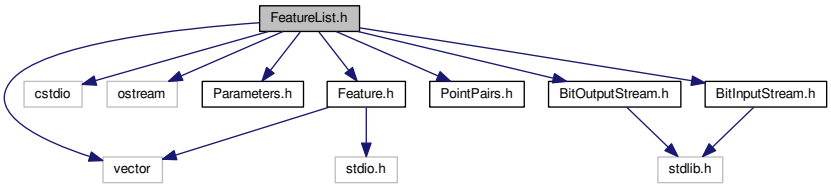
  *Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

## 8.10 FeatureList.h File Reference

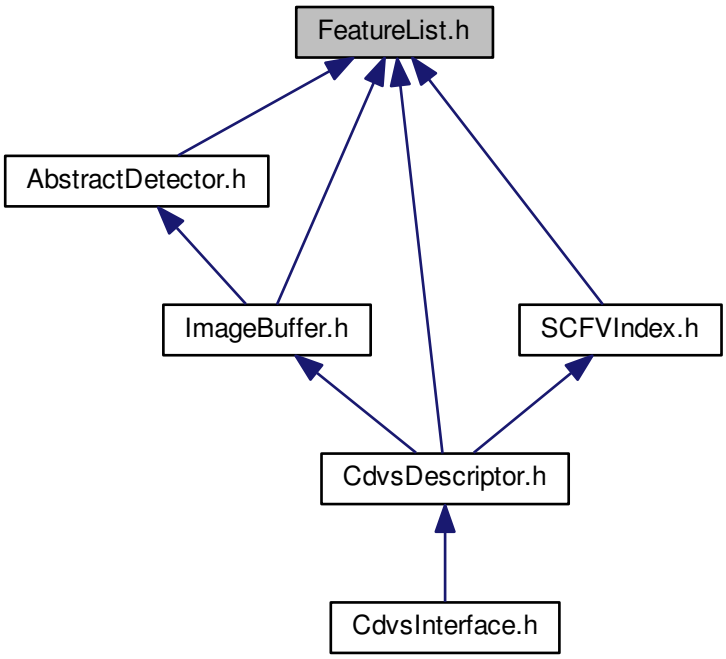
```
#include <vector>
```

```
#include <cstdio>
#include <ostream>
#include "Parameters.h"
#include "Feature.h"
#include "PointPairs.h"
#include "BitOutputStream.h"
#include "BitInputStream.h"
```

Include dependency graph for FeatureList.h:




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




## Data Structures

- class mpeg7cdvs::FeatureList

  *Container class for all features of an image.*
- class mpeg7cdvs::CompressedFeatureList

  *Container class for all compressed features of an image.*

### Namespaces

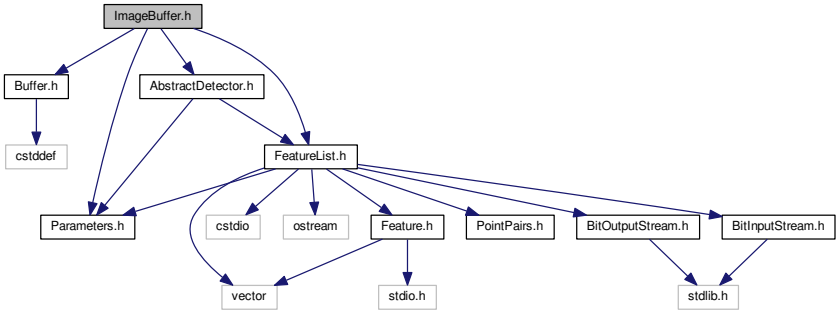
- mpeg7cdvs

  *Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

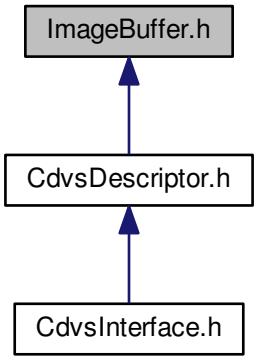
## 8.11 ImageBuffer.h File Reference

```
#include "Buffer.h"
#include "Parameters.h"
#include "FeatureList.h"
#include "AbstractDetector.h"
```

Include dependency graph for ImageBuffer.h:




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




### Data Structures

- class mpeg7cdvs::ImageBuffer

  *A container class for a bidimensional image; it's the base class of all keypoint detector classes.*

### Namespaces

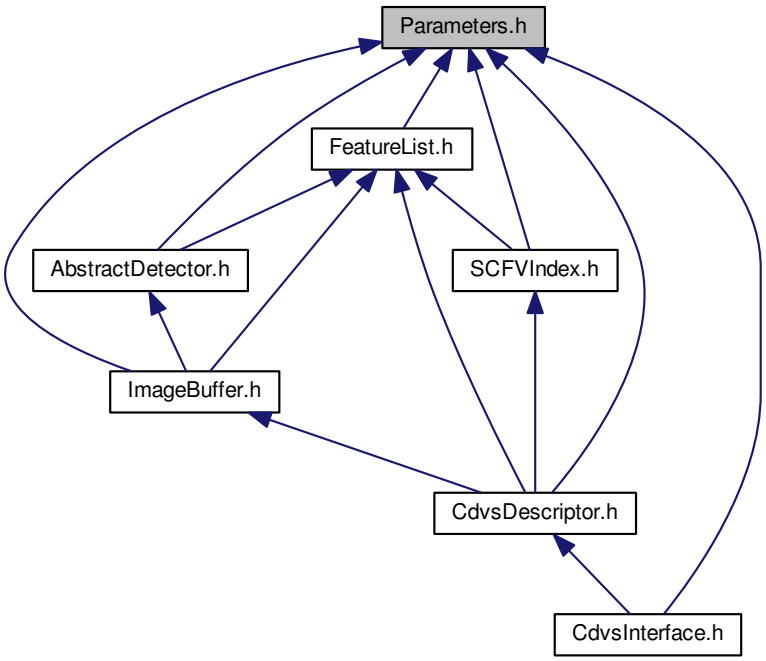
- mpeg7cdvs

  *Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

## 8.12 main.main File Reference

## 8.13 Parameters.h File Reference

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




### Data Structures

- class mpeg7cdvs::Parameters

  *Container for all encoding/decoding parameters associated to each target bitrate defined by MPEG CDVS.*

### Namespaces

- mpeg7cdvs

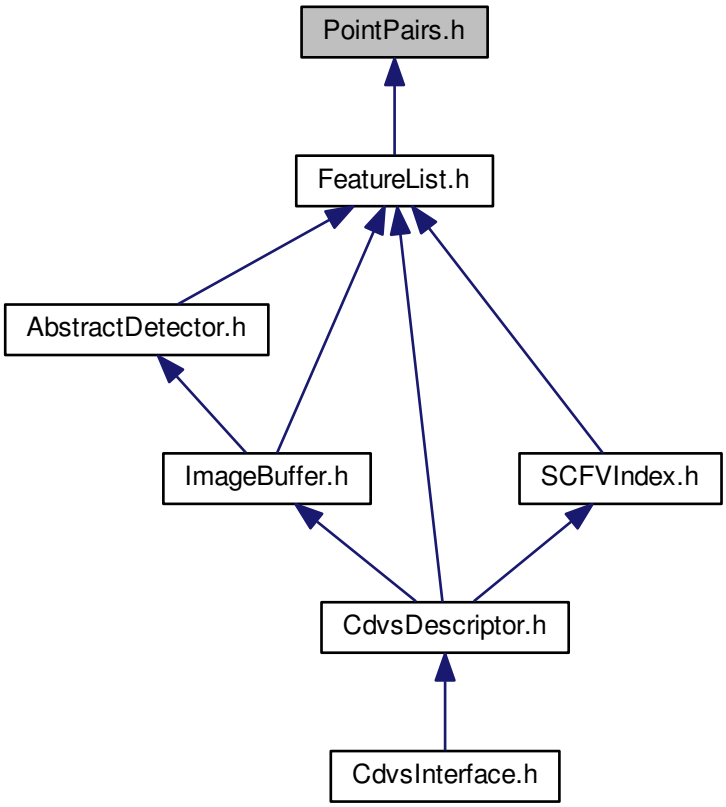
  *Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

### Typedefs

- typedef Parameters mpeg7cdvs::ParameterSet [Parameters::nModes]

## 8.14 PointPairs.h File Reference

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




### Data Structures

- class mpeg7cdvs::PointPairs

  *Parameter class, used to pass around matched point coordinates.*

### Namespaces

- mpeg7cdvs

  *Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

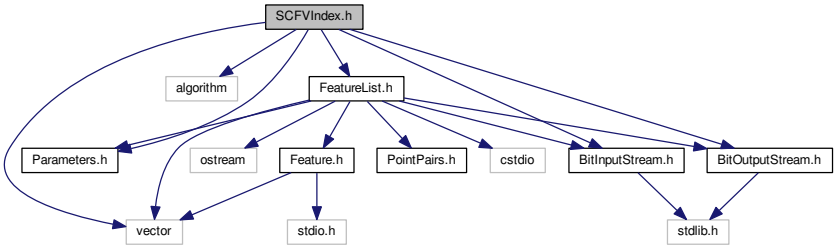
### Enumerations

- enum { mpeg7cdvs::match_2way_INTERSECTION = 0, mpeg7cdvs::match_2way_DISJOINT1 = 1, mpeg7cdvs::match_2way_DISJOINT2 = 2 }

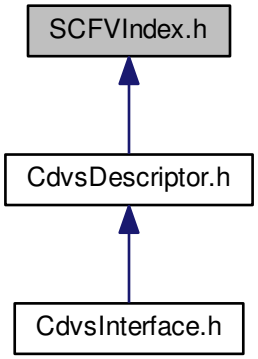
## 8.15 SCFVIndex.h File Reference

```
#include <vector>
#include <algorithm>
#include "FeatureList.h"
#include "Parameters.h"
#include "BitOutputStream.h"
#include "BitInputStream.h"
```

Include dependency graph for SCFVIndex.h:




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




### Data Structures

- class mpeg7cdvs::LookUpTable

  *A simple look up table implementation, to perform a bit count very quickly.*
- class mpeg7cdvs::SCFVSignature

  *Container class for a Scalable Fisher Vector binary signature; allows reading/writing from/to a bitstream, fetching/storing from/into a file, and comparing a signature with another.*
- class mpeg7cdvs::SCFVIndex

  *A class to manage an indexed list of SCFV signatures.*
- class mpeg7cdvs::SCFVFactory

  *A class to produce SCFV signatures.*

## Namespaces

- mpeg7cdvs

  *Namespace used to encapsulate all MPEG-7 CDVS declarations that are visible when the CDVS Library headers (in particular CdvsInterface.h) are included.*

## Variables

- static const float mpeg7cdvs::gama = 0.3f
- static const int mpeg7cdvs::num_bit_selection = 24
- static const int mpeg7cdvs::PCASiftLength = 32

  *number of principal components in the centroid space*
- static const int mpeg7cdvs::numberCentroids = 512

  *number of centroids of the codebook*

# Index