CDVS build and run instructions

1 Introduction

The CDVS Reference Software is composed by the CDVS Test Model, which includes the CDVS Library and a set of five executables, and by the CDVS Applications, which use the CDVS Library to perform some specific tasks (among which, conformance testing).
The CDVS Test model can be found in the *CDVS_evaluation_framework* directory, and must be installed first. The CDVS Applications can be found in the *CDVS_applications* directory.

2 Requirements

The code can be built on Linux or on Windows. In any case, we assume that the code will be build on 64 bit architectures, in order to be able to manage a large number of images; however, it is also possible to build the code on 32 bits systems for limited applications.
The CDVS code is entirely written in standard C and C++, and has been compiled on Windows 7 Enterprise 64-bit using Visual C++ 2010 (64 bit). The code can also be compiled on Linux 64-bit, as described below; in the following we provide a walk-through for Linux and Windows, respectively.

3 Building on Linux

This walkthrough has been performed on Ubuntu 14.04 LTS 64 bit (other distributions have similar package managers and libraries, but the exact name of each tool/package must be verified). On a clean installation of the system, the following tools and libraries must be installed before building the code:

- gcc/g++
- libtool
- autotools (automake, autoreconf, etc.)

On Ubuntu 14.04 this can be done using the following command:

```
sudo apt-get install build-essential autoconf libtool
```

Then, the *CDVS Test Model* and the *CDVS Applications* can be installed (in this order).

3.1 Building and installing the CDVS Test Model

The CDVS Test Model depends on the following libraries:

- Eigen
- JPEG
- FFTW (optional)

Install these libraries using the following command:

```
sudo apt-get install libjpeg-dev libeigen3-dev libfftw3-dev
```

To build the source code directly from the Subversion tree, go to the CDVS_evaluation_framework directory and run:

```
autoreconf --install --force
```

(this step is not needed if the package is distributed as a tarball source).

Then the usual *configure, make, make install* commands will build and install the code in /usr/local; however, in order to obtain a fast and optimized code, it is advisable to pass the following options to the configure script:

1. only if you will run the code on the same machine: -march=native;
2. only if you know that your CPU supports the POPCNT instruction: -mpopcnt;
3. to enable gcc/g++full optimization: -O2
4. to remove all debugging checks: -DNDEBUG
5. to enable multithreading: -fopenmp
6. to speed up compilation: -pipe

For example, you can define the following environment variable:

```
FLAGS="-march=native -mpopcnt -O2 -DNDEBUG -fopenmp -pipe"
```

and then run:

```
./configure CFLAGS="${FLAGS}" CXXFLAGS="${FLAGS}"
make
sudo make install
```

this will install the normative CDVS Library in /usr/local/lib and the CDVS Test Model executables in /usr/local/bin, while C++ file headers will be placed in /usr/local/include.

To install the non-normative variants of the CDVS_Library, (lowmem and bflog), run configure with the following options: --with-lowmem and --with-bflog respectively.
For example:

```
./configure --with-bflog --with-lowmem CFLAGS="${FLAGS}" \
CXXFLAGS="${FLAGS}"
make
sudo make install
```

will install an optimized build of all three variants of the CDVS Library in /usr/local/lib:

- libcdvs_main;
- libcdvs_bflog;
- libcdvs_lowmem;

and the following binaries in /usr/local/bin:

- extract;
- match;
- makeIndex;
- joinIndices;
- retrieve;

3.2 Building and installing CDVS Applications

The *CDVS_Applications* package contains applications that rely on a proper installation of the CDVS Library. Moreover, the HTTP server example depends on *libmicrohttpd;* this library can be installed on Debian/Ubuntu as follows:

```
sudo apt-get install libmicrohttpd-dev
```

To build the source code directly from the Subversion tree, go to the CDVS_applications directory and run:

```
autoreconf --install --force
```

Then running ./configure, make, make install will install in /usr/local/bin the following binaries:

- example1: encode and match two images;
- example2: retrieve one image from a DB;
- example3: retrieve multiple images from a DB;
- simpleextract: extraction of a CDVS descriptor from a JPEG image;
- simplematch: match and localize two CDVS descriptors;
- simpleretrieve: retrieve multiple images from a DB (similar to example 3);
- simpleextractbf: uses libcdvs_bflog in place of libcdvs_main (optional);
- parser: check the syntax of a CDVS descriptor;
- conformance: check CDVS conformance on multiple images/descriptors;
- httpserver: example HTTP server performing visual search;

Also in this case, it is possible to optimize the code using the following gcc/g++ options:

```
FLAGS="-march=native -O2 -mpopcnt"
```

Moreover, to install the optimized binaries in the local home directory it is possible to run configure as follows:

```
./configure --prefix=$HOME CFLAGS="${FLAGS}" CXXFLAGS="${FLAGS}"
make
make install
```

3.3 *Using a Visual Editor to build and debug the code*

The *CDVS Test Model* and the *CDVS Applications* projects can be built (in debug mode) and executed step-by-step using a Visual Editor (Eclipse):

1. download *Eclipse IDE for C/C++ Developers* from http://www.eclipse.org/downloads;
2. install it on your Linux PC (simply uncompress the tarball somewhere);
3. Run it;
4. select the default workspace directory;
5. close the welcome page, then click on File → Import... →Existing Projects into Workspace → Next → select the root directory of the SVN tree containing the CDVS Reference Software and import the two projects named "cdvs" and "CDVS_applications".

If the CDVS Library is already installed in the /usr/local directory, only the CDVS_applications project is needed to develop applications based on CDVS. In this case, uncheck the "cdvs" project before importing the existing projects.

4 Building the code on Windows

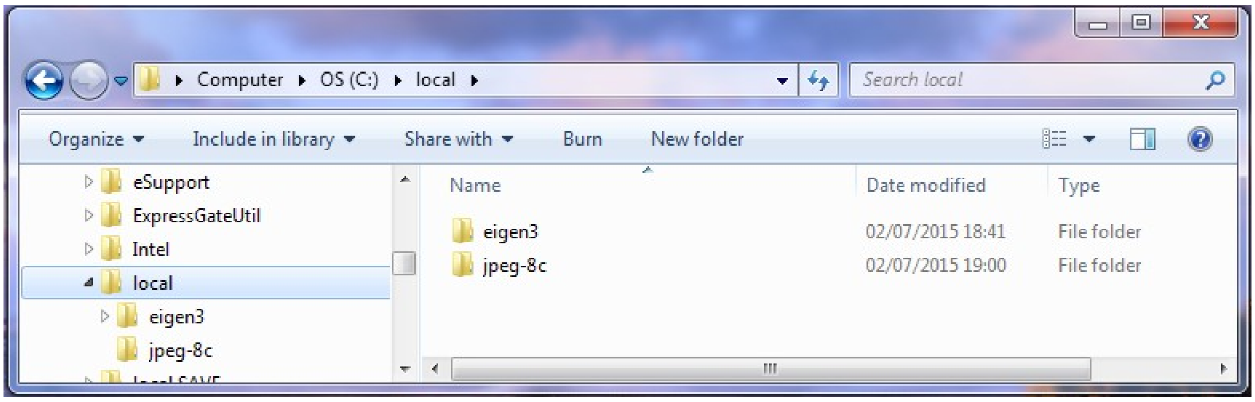
This walk-through has been performed on Windows 7 Enterprise 64-bit using Visual C++ 2010 (64 bit). The build configuration provided under Windows produces a subset of the binaries available under Linux (some optional binaries are not compiled). This allows to reduce the number of external libraries that must be downloaded and installed before compiling the code: in brief, installing the following libraries in C:\local will satisfy all dependencies:

1. Eigen
2. JPEG

(1) To install the Eigen library, go to http://eigen.tuxfamily.org/ and get the latest version (the current latest stable release is Eigen 3.2.5). Unzip the package into C:\local and rename the base directory of the library "eigen3".

(2) To install the JPEG library, download http://www.ijg.org/files/jpegsr8c.zip and unzip it into C:\local. Then, copy jconfig.vc into jconfig.h (in the jpeg-8c root directory). The library has no other dependencies.

When done, you should have the following directory structure:



Name	Date modified	Type
eigen3	02/07/2015 18:41	File folder
jpeg-8c	02/07/2015 19:00	File folder

Now you can open the CDVS_evaluation_framework/deployment/all_projects/all_projects.sln solution using Visual Studio 2010 (64-bit);

To build all applications, set:
- Solution Configurations: Release;
- Solution Platforms: x64;

and press F7 ("Build Solution"). This will compile all x64 applications and store the binaries in the "bin" directory.

The output results of the build process are the following binary executables in the CDVS_evaluation_framework/bin directory:
- extract.exe
- match.exe
- makeIndex.exe
- joinIndices.exe
- retrieve.exe

The same solution also builds the *CDVS Applications* on Windows; in this case, the output binaries will be stored in CDVS_applications/bin.

The output results of the build process are the following binary executables in the CDVS_applications/bin directory:

- conformance.exe
- simpleextract.exe
- simplematch.exe
- simpleretrieve.exe

5 Running the code

On Linux, if the `-march=native` option has been used, the code must be run on the same machine where it has been built. Also, code compiled on 64 bit machines can only run on 64 bit architectures. All applications are implemented as command line executables, and provide a usage message when run without parameters.

6 Documentation

Documentation of the CDVS Test Model and CDVS Library is available in the CDVS_evaluation_framework/docs directory of the SVN tree.