CDVA build and run instructions

1 Introduction

The CDVA Evaluation Framework is composed by a set of C++ classes that can be built on Linux and Windows to produce a single executable named *cdva*. The CDVA Evaluation Framework source code can be found in the *./src* directory.

2 Requirements

The code can be built on Linux or 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 video sequences; however, it is also possible to build the code on 32 bits systems for limited applications.
The CDVA code is entirely written in 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 walkthrough for Linux and Windows, respectively.

3 Building on Linux

This walkthrough has been performed on Ubuntu 16.04 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 16.04 this can be done using the following command:

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

Then, the *CDVA Experimentation Model* (CXM) can be installed.

3.1 Building and installing the CDVA framework

The CXM depends on the following libraries:

- opencv 2.4.x
- libcdvs_main (the CDVS library)

The opencv library can be installed using the following command (this will install opencv 2.4.9.1):

```
sudo apt install libopencv-dev
```

The CDVS library installation is described in the “CDVS build and run instructions” document.

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

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

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

The following adjustments to the CDVS evaluation framework installation are necessary.

Copy the following additional headers to the installation directory:

```
sudo cp ../CDVS_evaluation_framework/shared/ArithmeticCoding.h /usr/local/include/
sudo cp ../CDVS_evaluation_framework/shared/Match.h /usr/local/include/
```

Make the following change in CDVS_evaluation_framework/shared/FeatureList.cpp: remove inline for CompressedFeatureList::getDistance.

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. to enable g++ full optimization: -O3
2. to build in debugging mode: -g -O0
3. to speed up compilation: -pipe

For example, you can define the following environment variable:

```
FLAGS="-O3 -pipe"
```

and then run:

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

this will install the *cdva* executable in /usr/local/bin. If you prefer to install cdva in your own home directory instead, you can run configure as follows (no superuser privileges needed in this case):

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

this will install the *cdva* executable in $HOME/bin where $HOME is your own home directory.

3.2 Using a Visual Editor to build and debug the code

The CXM 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 CDVA Evaluation Framework and import the project named "CDVA_evaluation_framework".
6. Click on the "Build Project" icon.

4 Building the code on Windows

This walkthrough has been performed on Windows 7 Enterprise 64-bit using Visual C++ 2010 (64 bit).

To install it on Windows 64 bit,

1. download opencv v.2.4.11 from http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.11/opencv-2.4.11.exe/download
2. run the installation executable (it will unzip the package);
3. move the opencv directory to C:\local

Then set the following enviroment variable (this will hold the build directory of our OpenCV library that we use in our projects). Start up a command window as ADMINISTRATOR and enter:

```
setx -m OPENCV_DIR C:\local\opencv\build\x64\vc10    (for Visual Studio 2010 - 64 bit Windows)
setx -m OPENCV_DIR C:\local\opencv\build\x64\vc11    (for Visual Studio 2011 - 64 bit Windows)
setx -m OPENCV_DIR C:\local\opencv\build\x64\vc12    (for Visual Studio 2012 - 64 bit Windows)
```

Here the directory is where you have your OpenCV binaries (extracted or built). Inside this you should have two folders called lib and bin.
The -m should be added if you wish to make the settings computer wise, instead of user wise.

Then set the corresponding binary directory as an additional path (select the correct line):
PATH = %PATH%;C:\local\opencv\build\x64\vc10\bin
PATH = %PATH%;C:\local\opencv\build\x64\vc11\bin
PATH = %PATH%;C:\local\opencv\build\x64\vc12\bin

Then open the ".\build\win\VisualStudio_2010\cdva.sln" solution with Visual Studio, set:
- Solution Configurations: Release;
- Solution Platforms: x64;

and press F7 ("Build Solution"). This will produce the cdva executable.

5 Running the code

Open a command line shell ("cmd" on Windows, "Terminal" on Linux), locate the cdva executable and run it without parameters; it will print detailed usage information.

6 Documentation

Documentation of the CXM is available in “./docs“ directory of the SVN tree.