Installing open computer vision on windows

Youssef Hbali

August 27, 2013

Table of contents

Installing Git

Git is a distributed version control and source code management (SCM) system. We will use it to checkout opencv sources
Download git for windows

- Define a system variable GIT _HOME
- Add %GIT _HOME%\bin to the PATH system variable

Installing CMake

CMake is a cross-platform, open-source build system. CMake is a family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files. CMake generates native makefiles and workspaces that can be used in the compiler environment of your choice. *Download cmake-2.8.11.2-win32-x86.exe*

- Define a system variable CMAKE _HOME
- Add %CMAKE _HOME%\bin to the PATH system variable

Installing MinGW

MinGW, a contraction of "Minimalist GNU for Windows", is a minimalist development environment for native Microsoft Windows applications. *Download from http://www.mingw.org/*

- Define a system variable MinGW_HOME
- Add %MinGW_HOME%\bin to the PATH system variable

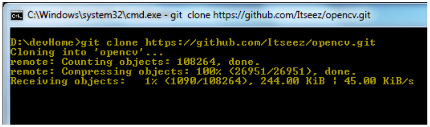
Installing Git

The Opencv project is hosted on github. To start a clone, go to *https://github.com/Itseez/opencv* and copy the HTTPS clone URL.

Getting OpenCv sources

Open a command line console and execute the following command to start cloning the project :

git clone https://github.com/Itseez/opencv.git



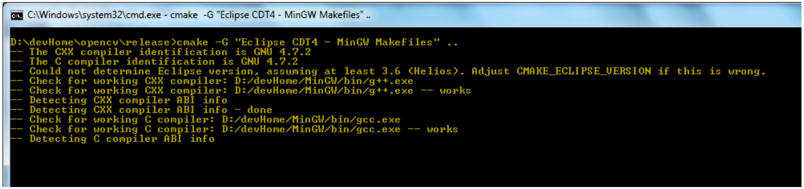
```
C:\Windows\system32\cmd.exe - git  clone https://github.com/Itseez/opencv.git

D:\devHome>git clone https://github.com/Itseez/opencv.git
Cloning into 'opencv'...
remote: Counting objects: 108264, done.
remote: Compressing objects: 100% (26951/26951), done.
Receiving objects:   1% (1090/108264), 244.00 KiB | 45.00 KiB/s
```

Getting OpenCv sources

For this tutorial, we will use opencv 2.4.4, so we will need to switch from the master branch to the 2.4.4 one, executing the following command : *git checkout 2.4.4*

Getting OpenCv sources

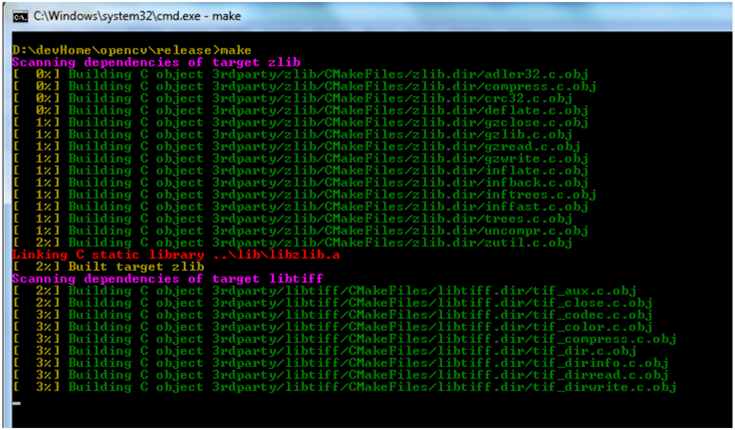
Time to start compilling opencv, it takes a little time. In the opencv directory that you have checketout, create a new directory called release for example. Get in the directory and execute the following command : *cmake -G "Eclipse CDT4 - MinGW Makefiles" ..*



```
C:\Windows\system32\cmd.exe - cmake  -G "Eclipse CDT4 - MinGW Makefiles" ..

D:\devHome\opencv\release>cmake -G "Eclipse CDT4 - MinGW Makefiles" ..
-- The CXX compiler identification is GNU 4.7.2
-- The C compiler identification is GNU 4.7.2
-- Could not determine Eclipse version, assuming at least 3.6 (Helios). Adjust CMAKE_ECLIPSE_VERSION if this is wrong.
-- Check for working CXX compiler: D:/devHome/MinGW/bin/g++.exe
-- Check for working CXX compiler: D:/devHome/MinGW/bin/g++.exe -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working C compiler: D:/devHome/MinGW/bin/gcc.exe
-- Check for working C compiler: D:/devHome/MinGW/bin/gcc.exe -- works
-- Detecting C compiler ABI info
```

Compilling the sources

To start the compilation, execute the following command : *make*

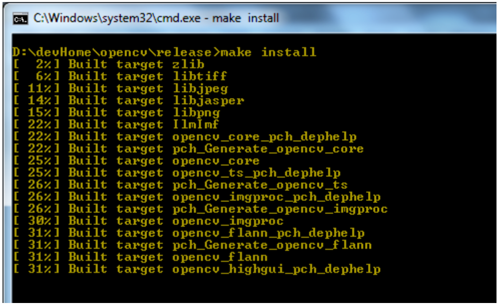


C:\Windows\system32\cmd.exe - make

```
D:\devHome\opencv\release>make
Scanning dependencies of target zlib
[  0%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/adler32.c.obj
[  0%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/compress.c.obj
[  0%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/crc32.c.obj
[  0%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/deflate.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/gzclose.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/gzlib.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/gzread.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/gzwrite.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/inflate.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/infback.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/inftrees.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/inffast.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/trees.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/uncompr.c.obj
[  2%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/zutil.c.obj
Linking C static library ..\lib\libzlib.a
[  2%] Built target zlib
Scanning dependencies of target libtiff
[  2%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_aux.c.obj
[  2%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_close.c.obj
[  3%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_codec.c.obj
[  3%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_color.c.obj
[  3%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_compress.c.obj
[  3%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_dir.c.obj
[  3%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_dirinfo.c.obj
[  3%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_dirread.c.obj
[  3%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_dirwrite.c.obj
```

Compilling the sources

To install the compiled programs, launch the following command :
make install



```
D:\devHome\opencv\release>make install
[  2%] Built target zlib
[  6%] Built target libtiff
[ 11%] Built target libjpeg
[ 14%] Built target libjasper
[ 15%] Built target libpng
[ 22%] Built target IlmImf
[ 22%] Built target opencv_core_pch_dephelp
[ 22%] Built target pch_Generate_opencv_core
[ 25%] Built target opencv_core
[ 25%] Built target opencv_ts_pch_dephelp
[ 26%] Built target pch_Generate_opencv_ts
[ 26%] Built target opencv_imgproc_pch_dephelp
[ 26%] Built target pch_Generate_opencv_imgproc
[ 30%] Built target opencv_imgproc
[ 31%] Built target opencv_flann_pch_dephelp
[ 31%] Built target pch_Generate_opencv_flann
[ 31%] Built target opencv_flann
[ 31%] Built target opencv_highgui_pch_dephelp
```

Configuring the project

- Create a new folder and named it DisplayImage
- Get in to this directory and create and directory named cmake-modules
- Download *FindOpenCV.cmake* and placed in the cmake-modules directory
- Edit the FindOpenCV.cmake file and add the following line to the begining of the file : set(OpenCV_DIR "D:/devHome/opencv/release")
- Go back to the DisplayImage folder and create a new file CMakeLists.txt with the following content :

Configuring the project

```
cmake_minimum_required (VERSION 2.6)
include_directories(${
   CMAKE_CURRENT_SOURCE_DIR}/include)
set(CMAKE_MODULE_PATH ${CMAKE_MODULE_PATH} $
   {CMAKE_CURRENT_SOURCE_DIR}/cmake-modules)

FIND_PACKAGE( OpenCV REQUIRED )
project (DisplayImage)
ADD_EXECUTABLE(DisplayImage display_image.
   cpp)
TARGET_LINK_LIBRARIES(DisplayImage ${
   OpenCV_LIBS})
```

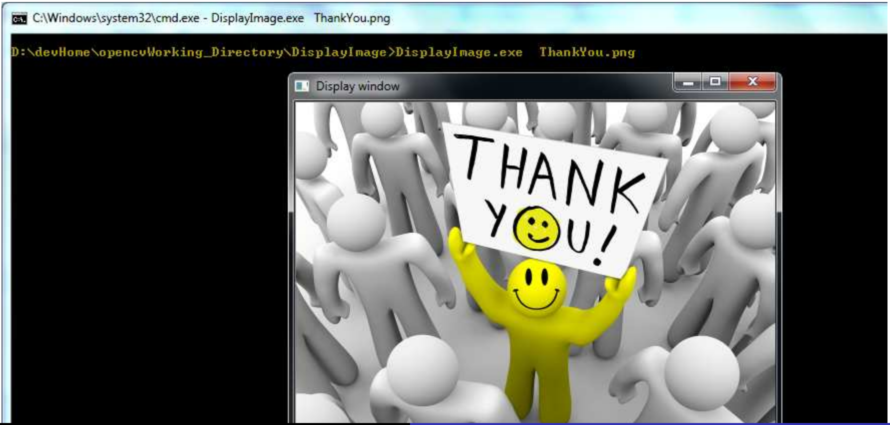
Configuring the project

Download The opencv dispaly image code example

Executing the example

To indicate to MinGW where to find the opencv dlls, make sure you have made the following :

- Define a system variable OPENCV _HOME
- Add %OPENCV _HOME%\bin to the PATH system variable



```
D:\devHome\opencvWorking_Directory\DisplayImage>DisplayImage.exe  ThankYou.png
```