Installing OpenFace from sources in Windows

Gabriel Vacaliuc

May 29, 2017

This document details the installtion of OpenFace[1] in Windows 10 from source files, without relying on the Visual Studio IDE.

[1] https://github.com/TadasBaltrusaitis/OpenFace

Environment Setup

CMake

CMake is a powerful build tool that will set up the correct makefiles for a slew of setups/IDEs/compilers. Likely the only change you'll have to make for everything in this guide to work with your environment is the `-G` flag to CMake.

This document will assume the use of CMake 3.8 (the latest as of compilation)[2] Navigate to the CMake Download page[3] and download the windows installer (*.msi).

[2] However, any CMake $>$ 3.0 or so should work fine.

[3] https://cmake.org/download/

During the installation, make sure to elect to add the execuatable path to the System PATH for (at least) the your profile.

Git

It's likely you already have Git installed, but in the event you don't, navigate to Git client download page[4] and download the 64-bit windows installer.

[4] https://git-scm.com/downloads

During the installation, make sure to elect to add the execuatable path to the System PATH for (at least) the your profile.

Microsoft Visual C++ Build Tools

For our C/C++ compiler, we'll install the MSVC 2017 Build Tools[5]. Note that if you already have Visual Studio 2017 installed, these will likely have already been installed. When installing, be sure to check the boxes to install the C++ compilers as well as C++/CLI Support.[6]

[5] https://www.visualstudio.com/downloads/#build-tools-for-visual-studio-2017

[6] Note that you don't have install the full VS IDE, we just need the build tools which are found on the same website.

After the installation, you should be able to navigate to the start menu and open both:

- Developer Command Prompt for VS 2017

- x64 Native Tools Command Prompt for VS 2017

For convenience, keep a single "x64 Native Tools Command Prompt for VS 2017" open for the duration of the installation. Depending on your setup, you may have to run it as administrator. This will allow us to set some environment variables to reduce typing.

Dependencies

zlib

The first dependency to install is zlib[7]. The sources can be downloaded from the URL to the right, making sure to download the *.zip.

[7] http://zlib.net

After downloading, extract the files to the directory `C:\local\`.[8] This should result in a directory such as `C:\local\zlib-1.x.x`.

[8] You may have to create this directory.

To build, navigate to your zlib directory and execute the following commands[9]:

[9] Note that this guide uses `nmake` as its build tool, however cmake provides generators for several IDEs, see output of `cmake -h` for details.

```
mkdir build
cd build
cmake .. -G "NMake Makefiles"
nmake
```

If everything worked correctly, you should have files `zlibd.lib`, `zlibd.dll` in your build directory. For convenience later, set some environment variables. Assuming that your zlib directory is `C:\local\zlib-1.2.11`, set:

```
set ZLIB_INCLUDE_DIR=C:\local\zlib-1.2.11
set ZLIB_LIBRARY=%ZLIB_INCLUDE_DIR%\build\zlibd.lib
```

libpng

Our second dependency depends directly on zlib. libpng can be downloaded from SourceForge[10], making sure to download the *.zip file.

[10] https://sourceforge.net/projects/libpng/files/libpng16/1.6.29/

After downloading, extract the files to `C:\local\`.

To build, navigate to the libpng directory: `C:\local\lpng1629\`, and execute:

```
mkdir build
cd build
```

```
cmake .. -DZLIB_LIBRARY=%ZLIB_LIBRARY% ^
        -DZLIB_INCLUDE_DIR=%ZLIB_INCLUDE_DIR% ^
        -G "NMake Makefiles"
mklink %ZLIB_INCLUDE_DIR%\zconf.h
    %ZLIB_INCLUDE_DIR%\zconf.h.in
nmake
```

If successful, you should the file `libpng16d.lib` in your build directory. For convenience later, set the following environment variables:

```
set PNG_INCLUDE_DIR=C:\local\lpng1629
set PNG_LIBRARY=%PNG_INCLUDE_DIR%\build\libpng16d.lib
```

libjpeg

The final dependency we'll build, libjpeg, is stored on github[11].

[11] https://github.com/stohrendorf/libjpeg-cmake.git

To download and build, execute the following in the ongoing Command Prompt from `C:\local\`:

```
git clone
    https://github.com/stohrendorf/libjpeg-cmake.git
    jpeg-9a
mkdir jpeg-9a\build
cd jpeg-9a\build
cmake .. -G "NMake Makefiles"
nmake
```

If successful, you should have the file `libjpeg.lib` in your build directory. For convenience later, set the following environment variables:

```
set JPEG_INCLUDE_DIR=C:\local\jpeg-9a
set JPEG_LIBRARY=%JPEG_INCLUDE_DIR%\build\libjpeg.lib
```

OpenCV

Download the OpenCV Windows installer from their SourceForge page[12]. Upon download, run the exe and install the files to `C:\local\`. For convenience later, set the following environment variable in your Command Prompt:

[12] https://sourceforge.net/projects/opencvlibrary/files/opencv-win/3.2.0/

```
set OPENCV_DIR=C:\local\opencv\build\x64\vc14\lib
```

Boost

DOWNLOAD the Boost Windows Setup from their SourceForge page[13]. Opt for the 64-bit version compiled with the latest version of MSVC (14.0). The filename you download should be:
`boost_1_59_0-msvc-14.0-64.exe`.

[13] https://sourceforge.net/projects/boost/files/boost-binaries/1.59.0/

Upon download, run the executable, and have it install the files to `C:\local`. For convenience later, set the following environment variables in your Command Prompt:

```
set BOOST_ROOT=C:\local\boost_1_59_0
set BOOST_LIBRARY_DIR=%BOOST_ROOT%\lib64-msvc-14.0
```

TBB

DOWNLOAD the prebuilt Windows libraries from Intel's TBB release page[14]. Be sure to download the file ending in '*win.zip'.

[14] https://github.com/01org/tbb/releases

EXTRACT the contents of the downloaded Zip File to `C:\local`. For convenience later, assuming that your extracted TBB directory is:
`C:\local\tbb2017_20170412oss`
set the following environment variables:

```
set TBB_ROOT_DIR=C:\local\tbb2017_20170412oss
set TBB_LIBRARY=%TBB_ROOT_DIR%\lib\intel64\vc14\tbb.lib
set TBB_INCLUDE_DIR=%TBB_ROOT_DIR%\include
```

OpenFace Installation

Header Fixing

CMake modifies the filename of a library header for libpng, so we'll create a symbolic link from the old filename to the true file. Assuming we have our environment variables set up, simply execute the commands:

```
mklink %PNG_INCLUDE_DIR%\pnglibconf.h
    %PNG_INCLUDE_DIR%\scripts\pnglibconf.h.prebuilt
```

Installation Proper

With all the dependencies in place, we can compile the OpenFace project. Working either in the original repository[15] or the forked

[15] https://github.com/TadasBaltrusaitis/OpenFace

repository[16] I'm providing, execute the following commands:

[16] https://github.com/gvacaliuc/OpenFace

```
mkdir build
cd build
# if using gvacaliuc/OpenFace
../build.bat
# else using TadasB/OpenFace
cmake .. ^
       -DOpenCV_DIR=%OPENCV_DIR% ^
       -DTBB_ROOT_DIR=%TBB_ROOT_DIR% ^
       -DTBB_LIBRARY=%TBB_LIBRARY% ^
       -DTBB_INCLUDE_DIR=%TBB_INCLUDE_DIR% ^
       -DZLIB_LIBRARY=%ZLIB_LIBRARY% ^
       -DZLIB_INCLUDE_DIR=%ZLIB_INCLUDE_DIR% ^
       -DPNG_PNG_INCLUDE_DIR=%PNG_INCLUDE_DIR% ^
       -DPNG_LIBRARY=%PNG_LIBRARY% ^
       -DJPEG_INCLUDE_DIR=%JPEG_INCLUDE_DIR% ^
       -DJPEG_LIBRARY=%JPEG_LIBRARY% ^
       -DCMAKE_BUILD_TYPE=Release ^
       -G "NMake Makefiles"
nmake
```

NOTE that the above commands assume that you've preserved the same Command Prompt that we defined the environment variables in. Note also that you must perform this compilation in an "x64 Native Tools Command Prompt for VS 2017", not a regular Developer Command Prompt.

Running the Executables

The executables that this guide builds has a few DLL dependencies that Windows needs to be aware of. In the Command Prompt you just built OpenFace in, update the `%PATH%` environment variable to include the OpenCV DLL, Boost DLL's, and the TBB DLL:

```
set TMP_OPENCV=%OPENCV_DIR%\..
set PATH=%PATH%;%TBB_ROOT_DIR%\bin\intel64\vc14
set PATH=%PATH%;%BOOST_LIBRARY_DIR%
set PATH=%PATH%;%TMP_OPENCV%\bin;
```

AFTER UPDATING the path, you should be able to run the executables from the `build\bin\Release` subdirectory like:

```
...\build\bin\Release> ..\FaceLandMarkImg.exe -root .
    -gaze -f input.png -oi output.png
```