

Tutorial for OpenCV 3.1.0 Installation on Raspberry Pi

Felipe Neiva Montalvão Melo, August 2016

Introduction

This tutorial goes through the installation process of OpenCV with brief explanation of the steps. It is based on the official OpenCV documentation, which can be easily found on the internet. Notice it is a installation guide specifically for **OpenCV version 3.1.0 on a Raspberry Pi running Raspbian OS**. However, this tutorial may be also taken as a reference on the installation of other versions of OpenCV on the Raspberry Pi or other platforms, since the basic process is the same. The main differences the versions and platforms cause on the process are simply related to the tools and the dependencies required. For more information on how to install OpenCV in other platforms, visit the webpages below:

Linux:

OpenCV 3.1.0 – Official Documentation, Aug 2016

http://docs.opencv.org/3.1.0/d7/d9f/tutorial_linux_install.html#gsc.tab=0

Since the Raspbian (The default Raspberry Pi OS) is a Linux system, this official installation tutorial for Linux should be very similar to the one presented in this document. This way, it can be considered as a second reference for the installation process.

Mac:

Non-official, Aug 2016

<http://blogs.wcode.org/2014/10/howto-install-build-and-use-opencv-macosx-10-10/>

Windows:

OpenCV 3.1.0 – Official Documentation, Aug 2016

http://docs.opencv.org/3.1.0/d3/d52/tutorial_windows_install.html#gsc.tab=0

Installation Process Overview

The OpenCV libraries are not simply installed by the use of a package manager neither by copying and pasting files to the desired directory on the computer. They are provided in the form of source code and therefore require compilation before being actually installed.

Just like a typical installation process on Linux, this one will be mainly executed through the use of commands on the terminal. Evidently, there are some parts that do not necessarily have to be executed by the terminal only. This tutorial, however, aims to make the process as fast/simplified as possible.

Before Starting

- **Make sure to have internet connection**

The installation process needs internet connection in order to work.

- **For installation over SSH**

It is recommended to use **X11** port forwarding if the OpenCV installation on the Raspberry Pi will be done through **SSH** since a graphical tool (**cmake-gui**) will be used to configure the installation. It is also possible to execute the console version of the cmake tool, however, it is much more complicated.

- **Working Directory**

In this guide, the main working directory will be the home folder but, of course, the user is free to choose any directory to work on. In order to get to the home folder and start the installation, simply open the terminal and input the command:

```
cd ~
```

Step 1: Dependencies & tools installation

The first step is to download/install the tools and dependencies. Simply input the following commands on the terminal.

[compiler]

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

[required]

```
sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev  
libavformat-dev libswscale-dev python-dev python-numpy
```

[optional]

```
sudo apt-get install libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev  
libjasper-dev libdc1394-22-dev
```

The optional packages are libraries/tools related to the applications the user intends to develop with OpenCV. For example, if the user wants support for png files, the package libpng-dev should be installed. **For a default installation, though, the code above should be just copied and, if any of the optional packages cannot be installed, it can be simply ignored.**

Step 2: Download OpenCV and extra modules source code

Using the commands `wget`, download the OpenCV 3.1.0 source code by inputting on the terminal:

```
wget https://github.com/Itseez/opencv/archive/3.1.0.zip
```

Also download OpenCV extra modules (contrib) from the github repository:

```
git clone https://github.com/opencv/opencv_contrib.git
```

Then unzip the OpenCV 3.1.0 downloaded file, move to the extracted folder and create a build folder inside of it:

```
unzip 3.1.0.zip
cd opencv-3.1.0/
mkdir build
```

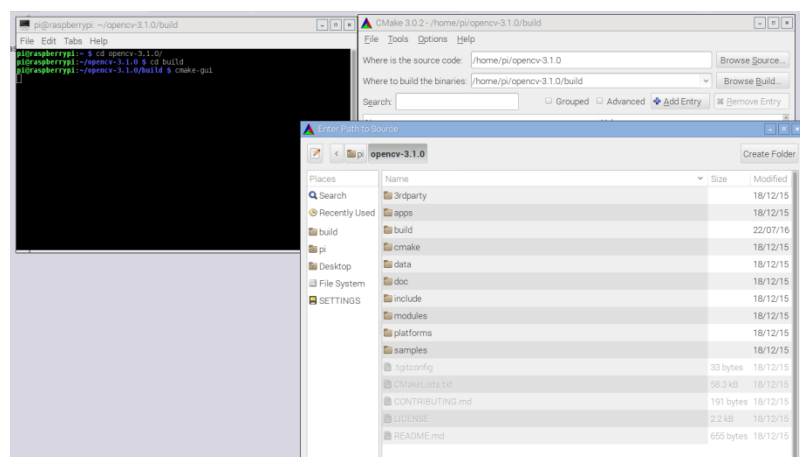
Move to the build folder:

```
cd build
```

Step 3: Configure OpenCV source code compilation using cmake (GUI)

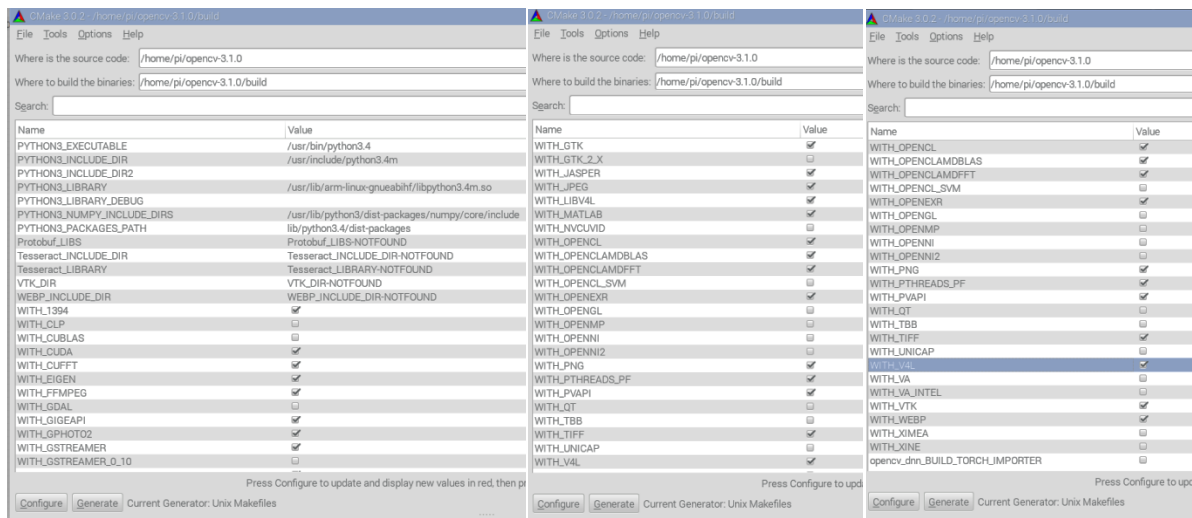
This step corresponds to the configuration of OpenCV files that are going to be installed. This is the most important and complicated part of the whole process. The user needs to carefully choose the modules that will be installed.

To do so, a graphical configuration tool (cmake) should be used. To start the tool, type `cmake-gui` on the terminal. Browse the location of the source codes and the build folder as well. Click on configure.



Selecting OpenCV source and build folder on cmake-gui

Find the entry `OPENCV_EXTRA_MODULES_PATH` and set it to the folder where the extra modules are located (`~/opencv_contrib/modules/`).



Module List

Click on configure button again once all of the modules are chosen. If all of the modules entries are not highlighted with red colour, the generate button can be finally clicked and the configuration process will be complete. Close the cmake-gui program and proceed to the next step.

****If the user for some reason is not able to use cmake-gui, the console version of the program can be used (cmake). This tutorial will not cover the use of the console version of cmake.***

Step 4: Compile and install

This part corresponds to the actual compilation of the OpenCV libraries and it may take a lot of time (up to 2 hours on a Raspberry Pi 3) to complete. After the compilation is complete, the generated files are moved to the installation folders and the process is finished. As the installation itself is basically moving files to the correct folders, it does not take much time (usually few seconds). Make sure the location on the terminal is the build directory.

Execute this code to **compile**:

```
make -j4
```

```
pi@raspberrypi: ~/opencv-3.1.0/build
File Edit Tabs Help

[ 10%] [ 10%] Built target opencv_test_imgproc_pch_dephep
Built target opencv_ml_pch_dephep
[ 10%] [ 10%] Built target opencv_photo_pch_dephep
Built target opencv_test_ml_pch_dephep
[ 10%] Built target opencv_perf_photo_pch_dephep
[ 10%] [ 10%] Generating opencv_perf_reg_pch_dephep.cxx
Generating opencv_reg_pch_dephep.cxx
[ 10%] Built target opencv_test_photo_pch_dephep
[ 10%] Generating opencv_test_reg_pch_dephep.cxx
Scanning dependencies of target opencv_reg_pch_dephep
Scanning dependencies of target opencv_perf_reg_pch_dephep
[ 10%] Building CXX object modules/reg/CMakeFiles/opencv_reg_pch_dephep.dir/ope
cv_reg_pch_dephep.cxx.o
Scanning dependencies of target opencv_test_reg_pch_dephep
[ 10%] Building CXX object modules/reg/CMakeFiles/opencv_perf_reg_pch_dephep.di
/opencv_perf_reg_pch_dephep.cxx.o
Linking CXX static library ../lib/libopencv_reg_pch_dephep.a
[ 10%] Building CXX object modules/reg/CMakeFiles/opencv_test_reg_pch_dephep.di
/opencv_test_reg_pch_dephep.cxx.o
[ 10%] Built target opencv_reg_pch_dephep
[ 10%] Generating opencv_surface_matching_pch_dephep.cxx
Scanning dependencies of target opencv_surface_matching_pch_dephep
[ 10%] Building CXX object modules/surface_matching/CMakeFiles/opencv_surface_ma
ching_pch_dephep.dir/opencv_surface_matching_pch_dephep.cxx.o
```

Console Output of make

Then execute this one to **install**:

```
sudo make install -j4
```

And finally update the system dynamic library linking configuration:

```
sudo ldconfig
```

Step 5: Check installation

To check if installation was successful, execute the two following commands:

1) `pkg-config --cflags opencv`

If installation is ok, this command should return

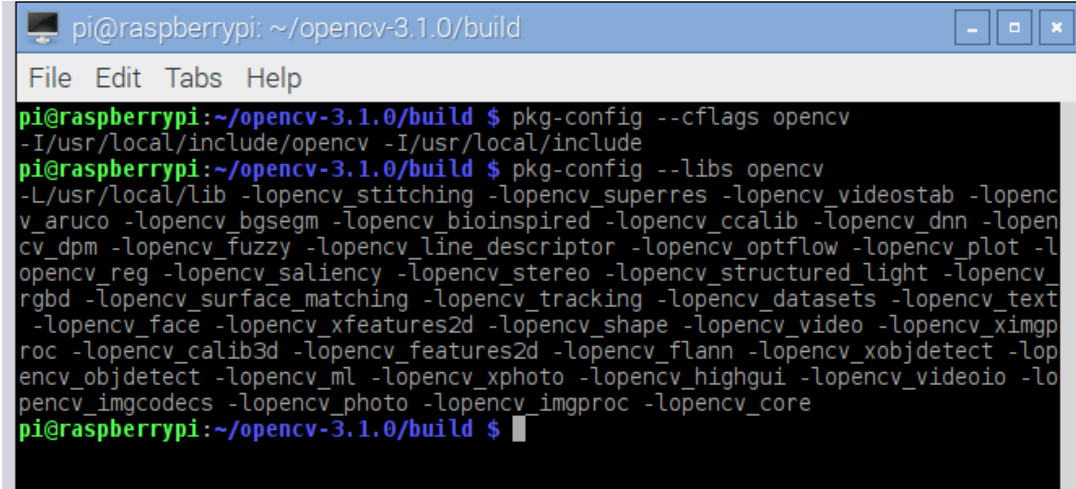
```
-I/usr/local/include/opencv -I/usr/local/include .
```

2) `pkg-config --libs opencv`

If installation is ok, this command should return

```
-L/usr/local/lib
```

and a list of installed libraries (main and extra modules).

A screenshot of a terminal window titled 'pi@raspberrypi: ~/opencv-3.1.0/build'. The terminal shows the execution of two pkg-config commands. The first command, 'pkg-config --cflags opencv', returns '-I/usr/local/include/opencv -I/usr/local/include'. The second command, 'pkg-config --libs opencv', returns a long list of library paths and names, including -L/usr/local/lib and various opencv_* modules like stitching, superres, videostab, aruco, bgsegm, bioinspired, ccalib, dnn, dpm, fuzzy, line_descriptor, optflow, plot, reg, saliency, stereo, structured_light, text, face, xfeatures2d, shape, video, ximgproc, calib3d, features2d, flann, xobjdetect, ml, xphoto, highgui, videoio, imgcodecs, photo, imgproc, and core. The prompt 'pi@raspberrypi:~/opencv-3.1.0/build \$' is visible at the bottom.

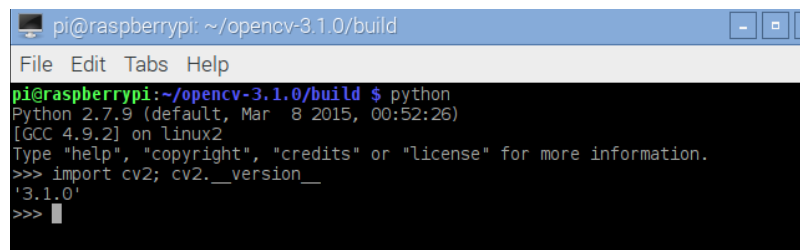
```
pi@raspberrypi: ~/opencv-3.1.0/build
File Edit Tabs Help
pi@raspberrypi:~/opencv-3.1.0/build $ pkg-config --cflags opencv
-I/usr/local/include/opencv -I/usr/local/include
pi@raspberrypi:~/opencv-3.1.0/build $ pkg-config --libs opencv
-L/usr/local/lib -lopencv_stitching -lopencv_superres -lopencv_videostab -lopenc
v_aruco -lopencv_bgsegm -lopencv_bioinspired -lopencv_ccalib -lopencv_dnn -loper
cv_dpm -lopencv_fuzzy -lopencv_line_descriptor -lopencv_optflow -lopencv_plot -l
opencv_reg -lopencv_saliency -lopencv_stereo -lopencv_structured_light -lopencv_
rgbd -lopencv_surface_matching -lopencv_tracking -lopencv_datasets -lopencv_text
-lopencv_face -lopencv_xfeatures2d -lopencv_shape -lopencv_video -lopencv_ximgp
roc -lopencv_calib3d -lopencv_features2d -lopencv_flann -lopencv_xobjdetect -lop
encv_objdetect -lopencv_ml -lopencv_xphoto -lopencv_highgui -lopencv_videoio -lo
pencv_imgcodecs -lopencv_photo -lopencv_imgproc -lopencv_core
pi@raspberrypi:~/opencv-3.1.0/build $
```

Pkg-config Console Output

Also, to check if OpenCV is correctly configured for **python**, open the python console by typing *python* on the terminal and then input the code:

```
import cv2; cv2.__version__
```

If it is installed correctly, this script should return '3.1.0' .



```
pi@raspberrypi: ~/opencv-3.1.0/build
File Edit Tabs Help
pi@raspberrypi:~/opencv-3.1.0/build $ python
Python 2.7.9 (default, Mar  8 2015, 00:52:26)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2; cv2.__version__
'3.1.0'
>>>
```

Python Console Output

Step 6: Compile C/C++ code

Compilation using command terminal

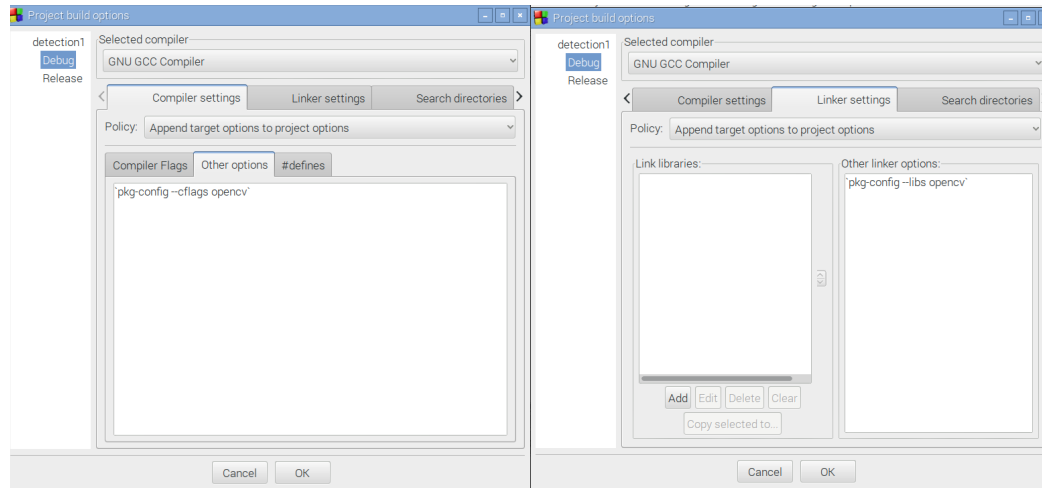
Example: If *source.cpp* is the C++ source code file and *program* is the compiled binary file

```
g++ `pkg-config --cflags opencv` source.cpp -o program `pkg-config --libs opencv`
```

Compilation using an IDE

Compiler flags: ``pkg-config --cflags opencv``

Linker Options: ``pkg-config --libs opencv``



Example of Project Configuration on Codeblocks IDE

*Notice that the symbols ``` are **grave accents** and not quotation marks. They should also be included when using the compilation/linkage parameters given above.

Uninstallation and Reinstallation

In order to uninstall the OpenCV simply navigate to the build folder and use the commands

```
sudo make uninstall
sudo ldconfig
```

The libraries that were compiled during the installation process will still be available in the build folder. This way, if the user wants to reinstall the OpenCV with the same libraries there is no need to recompile the source code again. The user can navigate to the build folder and use again the commands `sudo make install -j4` and `sudo ldconfig`.

If the user wants to add or remove a module on the system, the OpenCV first needs to be uninstalled and then installed again starting from step 3. If the build folder and its files were not deleted the compilation process will be faster. The files that had been already built (from the previous installation) will be skipped.

Appendix:

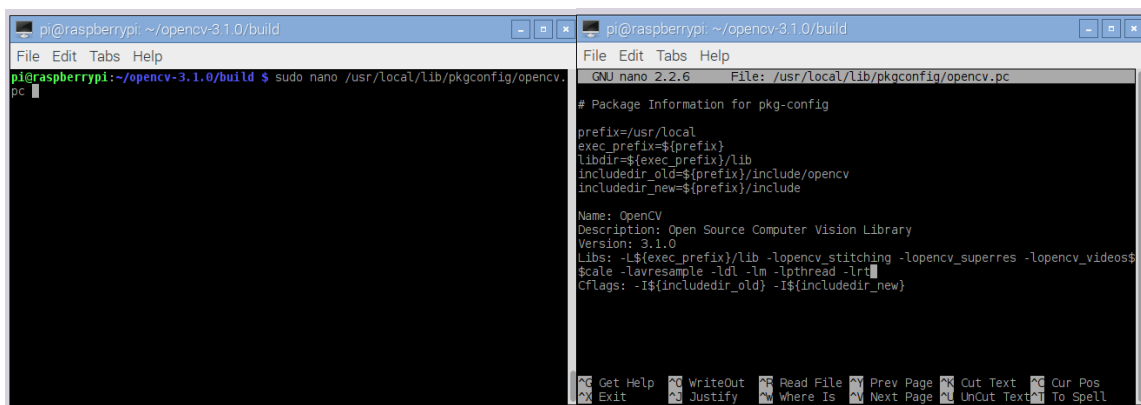
Default locations of of installed OpenCV

- **OpenCV libraries location:** `/usr/local/lib`
- **OpenCV headers location:** `/usr/local/include`; `/usr/local/include/opencv`

Problem with module linkage

If there is a linkage error during the compilation process of a C/C++ code that uses OpenCV it is certainly because the libraries that caused the error are not installed on the system. If the C/C++ code requires a library that is not present on the system OpenCV needs to be reinstalled.

However, if the user finds an error related to a library that is not being used on the code there is also the option to delete the linkage parameter in the file **`opencv.pc`** located at `/usr/local/lib/pkgconfig/`. By removing the name of the library that is causing the error from the **`opencv.pc`** file the user will completely **exclude** the library from the linkage process.



```
pi@raspberrypi: ~/opencv-3.1.0/build
pi@raspberrypi:~/opencv-3.1.0/build $ sudo nano /usr/local/lib/pkgconfig/opencv.pc
GNU nano 2.2.6 File: /usr/local/lib/pkgconfig/opencv.pc

# Package Information for pkg-config
prefix=/usr/local
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
includedir_old=${prefix}/include/opencv
includedir_new=${prefix}/include

Name: OpenCV
Description: Open Source Computer Vision Library
Version: 3.1.0
Libs: -I${exec_prefix}/lib -lopencv_stitching -lopencv_superres -lopencv_videostab
      -scale -lavresample -ldl -lm -lpthread -lrt
CFlags: -I${includedir_old} -I${includedir_new}
```

Editing the file `opencv.pc` using nano text editor