Installation Cheat Sheet 4 - OpenCV 3 Compile From Source
Including Configuration with Visual Studio and Qt
Using Windows 10 + Visual Studio 2013 or Visual Studio 2015 or Qt 5.X.X

Click here to go to the YouTube video for this Cheat Sheet

GitHub page with all Cheat Sheets and code

If you found this Cheat Sheet helpful please consider supporting me on Patreon

Note: **Bold blue** indicates something that will change depending on your version of OpenCV, CMake, Visual Studio, or Qt

----
Table of Contents

Part I - OpenCV Compile From Source

1a) Download and install Visual Studio **2013** or **2015** Community Edition (yes, both are free, and choosing all the default options will work fine)
1b) Note: As of when OpenCV **3.0.0** was released Visual Studio **2015** was not yet available and therefore Visual Studio **2015** could not be tested with OpenCV **3.0.0** before release. Also, as of the time this is being written the most current build of Qt (**5.5.0**) supports Visual Studio **2013** but not **2015**. Therefore if you are using OpenCV **3.0.0** and Qt **5.5.0** it is recommended to perform the remaining steps in this tutorial with Visual Studio 2013. If you are using any version of OpenCV after **3.0.0** and a **later version of Qt that supports Visual Studio 2015** then either Visual Studio **2013** or **2015** will work.

2a) Download the latest 32-bit version of Qt for your version of Visual Studio, for example:
"qt-opensource-windows-x86-msvc**2013**-**5.5.0**.exe"
(Note: Do NOT use the Qt Online Installer, this will download the 64-bit version if you have a 64-bit computer)
(Note 2: On the Qt page when you get to "Download Now", scroll down further to "View All Downloads", this should list the Offline Installer for the 32-bit MSVC version of Qt as a choice)
2b) Install Qt (choose default options)

3a) Download the latest version of OpenCV, for example OpenCV **3.0.0**
3b) Make a folder "C:\OpenCV-**X.X.X**" for your version of OpenCV, ex. "C:\OpenCV-**3.0.0**" and extract OpenCV to there
3c) Within "C:\OpenCV-**X.X.X**" make a folder "\mybuild", ex make a folder "C:\OpenCV-**3.0.0**\mybuild"

4a) Download and install the latest version of CMake with the Windows 32-bit Installer, for example: "cmake-**3.3.1**-win32-x86.exe"
4b) If the CMake installation hangs before it gets started, reboot, then instead of double-clicking on the CMake download, right-click on the CMake download and choose "Run as administrator"
4c) During the CMake installation, choose the option "Add CMake to the system PATH for all users" (for all the other options the defaults are ok)
4d) Reboot

5a) In File Explorer, navigate to the Qt bin directory which should have various Qt DLLs, ex navigate to "C:\Qt\Qt**5.5.0****5.5**\msvc**2013**\bin", verify that various Qt DLLs are in this directory (ex. Qt5Cored.dll, Qt5Widgetsd.dll, etc.)
5b) Add the Qt bin directory to your path, ex. add "C:\Qt\Qt**5.5.0****5.5**\msvc**2013**\bin" to PATH
5c) Reboot
(Note: this step is necessary for CMake to find the Qt DLLs while compiling OpenCV, and also for OpenCV programs to find the Qt DLLs when running, do not skip this step!!)

6a) Start CMake
6b) Set "Where is the source code:" to your OpenCV sources directory, ex "C:/OpenCV-**3.0.0**/opencv/sources"
6c) Set "Where to build the binaries:" to your OpenCV mybuild directory, ex: "C:/OpenCV-**3.0.0**/mybuild"
6d) Press "Configure"
6e) Choose "Visual Studio **12 2013**" or "Visual Studio **14 2015**" as applicable from the drop-down menu (do NOT choose the 64-bit option, which is titled "Visual Studio **XX** 201**X** Win64"), choose the "Use default native compilers" radio button, then choose "Finish"
6f) After a moment you will get a list of options, all in red. Scroll towards the bottom and check "WITH_QT", then press "Configure" again.
6g) After another moment the previous lines should now be white, with new lines pertaining to Qt only in red. Press "Configure" a 3rd time, after a moment all lines should now be white.
6h) Press "Generate"
6i) When generating is done, in your OpenCV mybuild directory, ex. "C:\OpenCV-**3.0.0**\mybuild" there should be a file "OpenCV.sln", this is a regular Visual Studio solution file

7a) Double click "OpenCV.sln" to open in Visual Studio, verify "Solution Configurations" and "Solution Platforms" are set to "Debug" and "Win32" respectively, then choose "Build -> Build Solution"
7b) Compiling will take at least a few minutes (OpenCV is a huge program)
7c) When compiling is complete you will likely get multiple warnings and a linker error pertaining to "python27_d.lib", as long as the error(s) only pertain to Python it's ok

8a) Remove any OpenCV directories in your PATH currently, for example if you added "C:\OpenCV-**X.X.X**\opencv\build\x86\vc**XX**\bin" when following part 1 then remove that at this time
8b) After completing the compile from source in the previous step, the compiled from source OpenCV **bin** directory should be located at:
C:\OpenCV-**X.X.X**\mybuild\bin\Debug
Verify that this directory contains the successfully compiled OpenCV DLLs (ex. opencv_calib3d**300**d.dll,

opencv_core**300**d.dll, opencv_features2d**300**d.dll, etc), then add this directory to your PATH
8c) Pull up a Command Prompt and verify the OpenCV bin directory, and the Qt bin directory from step 5, are now in PATH, then reboot
(note that the **bin** directory is *different* from the precompiled binary directory used in part 1)

Part II - Configuration with Visual Studio

1) From my MicrocontrollersAndMore GitHub page decide which example you are going to use:
CannyStill.cpp (uses a still image)
CannyWebcam.cpp (uses a webcam)
RedBallTracker.cpp (tracks a red ball, uses a webcam)

2a) Start Visual Studio, make a new project
2b) Choose Visual C++, Win32 Console Application, name as you prefer, ex "CannyStill2", set preferred location, uncheck "Create directory for solution" and "Add to source control", choose OK
2c) On the **"Welcome to the Win32 Application Wizard"** screen choose Next
2d) On the **"Application Settings"** screen, uncheck "Precompiled Header" and "Security Development", check "Empty Project", and verify "Console application" radio button is checked, then choose Finish

3a) Right click in Solution Explorer, choose Add -> New Item
3b) Choose "C++ File", name the C++ file as preferred, ex. "CannyStill2.cpp"
3c) Copy/paste the entire code from your chosen example into the .cpp file
(At this point Visual Studio will underline many of the lines of code with red because we have not yet informed Visual Studio as to the location of OpenCV, subsequent steps will resolve this)

4) If you are using an example with a still image (i.e. CannyStill.cpp), copy any JPEG image into the project directory and rename it "image.jpg". You can use the "image.jpg" from my MicrocontrollersAndMore GitHub page if you would like to see the same results as in the video (if you are using a webcam example then this step does not apply).

5) In VS go to:
Project -> Properties -> Configuration Properties -> VC++ Directories -> Include Directories
add the **include** directory, ex: C:\OpenCV-**3.0.0**\opencv\build\include
(note that the **include** directory is *the same as* the precompiled binary directory used in part 1)

6) In VS go to:
Project -> Properties -> Configuration Properties -> VC++ Directories -> Library Directories:
add the **library** directory, ex: C:\OpenCV-**3.0.0**\mybuild\lib\Debug
(note that the **library** directory is *different* from the precompiled binary directory used in part 1)

7) In File Explorer (not within Visual Studio) navigate to the **lib** directory, ex:
C:\OpenCV-**3.0.0**\mybuild\lib\Debug
In the lib directory you will find the debug libs (ending with a 'd'), for example if you are using OpenCV **3.0.0** the debug libs are the following:

opencv_imgcodecs**300**d.lib
opencv_imgproc**300**d.lib
opencv_ml**300**d.lib
opencv_objdetect**300**d.lib
opencv_photo**300**d.lib
opencv_shape**300**d.lib
opencv_stitching**300**d.lib
opencv_superres**300**d.lib
opencv_ts**300**d.lib
opencv_video**300**d.lib
opencv_videoio**300**d.lib
opencv_videostab**300**d.lib
opencv_calib3d**300**d.lib
opencv_core**300**d.lib
opencv_features2d**300**d.lib
opencv_flann**300**d.lib
opencv_hal**300**d.lib
opencv_highgui**300**d.lib

Note that the "**300**" corresponds to OpenCV **3.0.0**, these will be different if you are using a different version of OpenCV, also which libraries are present may vary with OpenCV version.

Copy/paste each of these names into the following location in Visual Studio:
Project -> Properties -> Configuration Properties -> Linker -> Input -> Additional Dependencies

8) In the Visual Studio toolbar, verify that "Solution Configurations" and "Solution Platforms" are set to "Debug" and "Win32", respectively

9) Run the program, either without debugging (choose Debug, then the hollow green arrow, or press Ctrl+F5) or with debugging (solid green arrow or press F5)

Part III - Configuration with Qt for a Non-GUI Program

1) Part I is a prerequisite to Part III. If you have not already performed all steps from Part I please return to Part I and do so before continuing

2a) From my MicrocontrollersAndMore GitHub page decide which example you are going to use:
CannyStill.cpp (uses a still image, no Qt GUI, same program as in Installation Cheat Sheet 1)
CannyStillWithQtGUI.cpp (uses a still image and a Qt GUI)
CannyWebcamWithQtGUI.cpp (uses a webcam and a Qt GUI)
RedBallTrackerWithQtGUI.cpp (tracks a red ball, uses a webcam and a Qt GUI)
2b) If you are interested in an example with a Qt GUI, scroll down to Part IV. If you are interested in an example without a Qt GUI, continue here.

3) Start Qt Creator, choose New Project

4) On the **"New Project"** screen, choose "Non-Qt Project" and "Plain C++ Project", then "Choose..."

5a) On the **"Introduction and Project Location"** screen, choose Name as preferred, for example "CannyStill1"
5b) Choose your preferred "Create in:" location, for example "C:\QtProgs"
5c) Check "Use as default project location" if you plan on using the chosen location for future Qt programs
5d) Choose "Next >"

6a) On the **"Kit Selection"** screen press "Details" to show all builds
6b) Uncheck options until only one is remaining, the Debug build for the 32-bit compile with MSVC 201**X**
6c) Set the "Debug" directory/name to be the combination of the directory/name from the previous screen, for example, set "Debug" to "C:\QtProgs\CannyStill1". If this is set to the same directory/name as the directory/name from the previous screen your build will be included in the same directory as your project files. If you do *not* set the location and name to match the previous screen, Qt will create separate project and build directories, this may cause confusion and is *not* recommended.
6d) Choose "Next >"

7a) On the **"Project Management"** screen, verify "Add to version control:" is set to "<None>"
7b) choose "Finish"

8) Go to "main.cpp", copy/paste the code from your chosen example
(At this point Qt Creator will underline your includes because we have not yet informed Qt Creator as to the location of OpenCV, subsequent steps will resolve this)

9) In Windows File Explorer, navigate to your OpenCV lib directory, ex "C:\OpenCV-**3.0.0**\mybuild\lib\Debug", then make a list of all filenames that end "d.lib". Next in Qt Creator, go to your .pro file, ex "CannyStill1.pro", and add the following, making sure to change the libraries per what is present in your lib directory depending on your version of OpenCV:

```
################################################################################
# add these to the end of your .pro file, this is so Qt knows about the location of the include and lib directories
# in Qt .pro files, begin a line with a pound character '#' to enter a comment
# note that for the double backslashes, the second is an escape character so the first is seen by Qt as a backslash
# the single backslashes at the end of each line (except for the last line) are line continuation characters

INCLUDEPATH += C:\\OpenCV-3.0.0\\opencv\\build\\include

LIBS += -LC:\\OpenCV-3.0.0\\mybuild\\lib\\Debug \
    -lopencv_imgcodecs300d \
    -lopencv_imgproc300d \
    -lopencv_ml300d \
    -lopencv_objdetect300d \
    -lopencv_photo300d \
    -lopencv_shape300d \
    -lopencv_stitching300d \
    -lopencv_superres300d \
    -lopencv_ts300d \
    -lopencv_video300d \
    -lopencv_videoio300d \
    -lopencv_videostab300d \
    -lopencv_calib3d300d \
    -lopencv_core300d \
    -lopencv_features2d300d \
    -lopencv_flann300d \
    -lopencv_hal300d \
    -lopencv_highgui300d

# Note: it is recommended to add a blank line at the end of your .pro file ##############################
```

The above was for OpenCV **3.0.0**, the libraries will be named differently and different libraries may be present in the lib directory in future versions of OpenCV, make sure to change the above accordingly.

10) If you are using an example with a still image (i.e. CannyStill.cpp), copy any JPEG image into the project directory and rename it "image.jpg". You can use the "image.jpg" from my MicrocontrollersAndMore GitHub page if you would like to see the same results as in the video (if you are using a webcam example then this step does not apply).

11a) Run the program by clicking on the applicable icon in the lower left corner, you can choose either "Run" (green arrow) or "Start Debugging" (green arrow with a bug on top)

11b) If upon attempting to run the program, you receive unusual or illogical errors, ex "Cannot open include file: 'opencv2/core/core.hpp': No such file or directory" and you are certain you have set everything up correctly, as a first troubleshooting step, choose ***Build -> Clean All***, then ***Build -> Run qmake***, then attempt to run your program again.

Part IV - Configuration with Qt for a GUI Program

1) Part I is a prerequisite to Part IV. If you have not already performed all steps from Part I please return to Part I and do so before continuing.

2) Start Qt Creator, choose New Project

3) On the **"New Project"** screen, choose "Application" and "Qt Widgets Application", then "Choose . . ."

4a) On the **"Introduction and Project Location"** screen, choose Name as preferred, for example "CannyStillWithQtGUI1"

4b) Choose your preferred "Create in:" location, for example "C:\QtProgs"
4c) Check "Use as default project location" if you plan on using the chosen location for future Qt programs
4d) Choose "Next >"

5a) On the **"Kit Selection"** screen press "Details" to show all builds
5b) Uncheck options until only one is remaining, the Debug build for the 32-bit compile with MSVC 201**X**
5c) Set the "Debug" directory/name to be the combination of the directory/name from the previous screen, for example, set "Debug" to "C:\QtProgs\ CannyStillWithQtGUI1". If this is set to the same directory/name as the directory/name from the previous screen your build will be included in the same directory as your project files. If you do *not* set the location and name to match the previous screen, Qt will create separate project and build directories, this may cause confusion and is *not* recommended.
5d) Choose "Next >"

6a) On the **"Class Information"** screen, choose "Class name:" as preferred, for example "frmMain", note this sets the name of "Header file:", "Source file:" and "Form file:" for you. It is NOT recommended to change the name of "Header file:", "Source file:", or "Form file:" directly
6b) Verify "Base class:" is set to QMainWindow and verify "Generate form:" is checked
6c) choose "Next >"

7a) On the **"Project Management"** screen, verify "Add to version control:" is set to "<None>"
7b) choose "Finish"

8) In Windows File Explorer, navigate to your OpenCV lib directory, ex "C:\OpenCV-**3.0.0**\mybuild\lib\Debug", then make a list of all filenames that end "d.lib". Next, in Qt Creator, go to your .pro file, ex "CannyStill1.pro", and add the following, making sure to change the libraries per what is present in your lib directory depending on your version of OpenCV:

```
##########################################################################
# add these to the end of your .pro file, this is so Qt knows about the location of the include and lib directories
# in Qt .pro files, begin a line with a pound character '#' to enter a comment
# note that for the double backslashes, the second is an escape character so the first is seen by Qt as a backslash
# the single backslashes at the end of each line (except for the last line) are line continuation characters

INCLUDEPATH += C:\\OpenCV-3.0.0\\opencv\\build\\include

LIBS += -LC:\\OpenCV-3.0.0\\mybuild\\lib\\Debug \
    -lopencv_imgcodecs300d \
    -lopencv_imgproc300d \
    -lopencv_ml300d \
    -lopencv_objdetect300d \
    -lopencv_photo300d \
    -lopencv_shape300d \
    -lopencv_stitching300d \
    -lopencv_superres300d \
    -lopencv_ts300d \
    -lopencv_video300d \
    -lopencv_videoio300d \
    -lopencv_videostab300d \
    -lopencv_calib3d300d \
    -lopencv_core300d \
    -lopencv_features2d300d \
    -lopencv_flann300d \
    -lopencv_hal300d \
    -lopencv_highgui300d

# Note: it is recommended to add a blank line at the end of your .pro file ##############################
```

The above was for OpenCV **3.0.0**, the libraries will be named differently and different libraries may be present in the lib directory in future versions of OpenCV, make sure to change the above accordingly.

9) On the left choose "Design" or double click on your form file, ex "frmmain.ui", this will bring up the form editor

If this is your 1st time using Qt, you can change form design options by going to:
Tools -> Options -> Designer (toward the lower left)
I recommend changing the grid to something much smaller than the default, for example 4 x 4

10a) In the Object Inspector window (towards the top right once you are in the Form Editor), right click on "menuBar QMenuBar" and choose "Remove Menu Bar"
10b) Perform the same steps to remove "mainToolBar QToolBar" and "statusBar QStatusBar"
(Note: If you plan on using the QMenuBar, QToolBar, or QStatusBar for something later on do not remove them, however for any of the example programs in this installation guide removal of these is recommended)

11a) Depending on which of the above examples you are using, add the applicable widgets to the form (found in the comments section at the top of the source). For example if you are using "CannyStillWithQtGUI.cpp", add the following widgets:
btnOpenFile
lblChosenFile
lblOriginal
lblCanny
11b) Set widget properties as shown in the video or as desired

12a) For any button with an associated event, ex btnOpenFile, right click on the button and choose "Go to slot..."
12b) Choose "clicked()", then OK (this will write the beginning of the button event for you)
(this step does not apply if your chosen example does not have a button)

13) Copy/paste the remaining portions of the code only (do NOT paste over any code written by Qt Creator) from your chosen example

14a) Run the program by clicking on the applicable icon in the lower left corner, you can choose either "Run" (green arrow) or "Start Debugging" (green arrow with a bug on top)

14b) If upon attempting to run the program, you receive unusual or illogical errors, ex "Cannot open include file: 'opencv2/core/core.hpp': No such file or directory" and you are certain you have set everything up correctly, as a first troubleshooting step, choose ***Build -> Clean All***, then ***Build -> Run qmake***, then attempt to run your program again.