Installation Cheat Sheet 4 - OpenCV 2.4.11 Compile From Source and Qt using Windows 7 + Qt ???

(should also work with Windows 8/8.1, not tested though)

download and install Visual Studio 2013 Community Edition (yes, its free) (choosing all default options will work fine)

download the latest 32-bit version of Qt, for example:
qt-opensource-windows-x86-msvc2013_opengl-5.4.1.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 Windows Visual Studio compiler version of Qt as a choice)

download OpenCV 2.4.11

make a folder "C:\OpenCV-2.4.11" and extract OpenCV 2.4.11 to there
make a folder "C:\OpenCV-2.4.11\mybuild"

download the latest version of CMake for windows with the Windows 32-bit Installer,
for example "cmake-3.2.2-win32-x86.exe"

during the CMake installation, choose the option "Add CMake to the system PATH for all users"

reboot, start CMake

set "Where is the source code:" to C:/OpenCV-2.4.11/opencv/sources
set "Where to build the binaries:" to C:/OpenCV-2.4.11/mybuild

press Configure, choose "Visual Studio 12 2013" from the drop-down menu
(do NOT choose the 64-bit option, which is titled "Visual Studio 12 2013 Win64")
choose the "Use default native compilers" radio button, then choose Finish
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
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
press Generate

when generating is done, in "C:\OpenCV-2.4.11\mybuild" there should be a file "OpenCV.sln"
this is a regular Visual Studio 2013 solution file, double click to open in Visual Studio
verify "Solution Configurations" and "Solution Platforms" are set to "Debug" and "Win32" respectively,
then choose:
Build -> Build Solution

you will likely get multiple warnings and a linker error pertaining to "python27_d.lib",
as long as there are no other errors its ok

remove any OpenCV directories in your PATH currently, for example if you added
"C:\OpenCV-2.4.11\opencv\build\x86\vc12\bin" when following part 1 then remove that at this time

add the updated **bin** directory to the operating system PATH:
C:\OpenCV-2.4.11\mybuild\bin\Debug
pull up Command Prompt and verify the updated **bin** directory (and no other OpenCV directories) is now in PATH, then reboot
(note that the **bin** directory is *different* from the precompiled binary directory used in part 1)

instructions for using the compile from source within Visual Studio (scroll down further if in interested in Qt):

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)

start Visual Studio 2013, make a new project
choose Visual C++, Win32 Console Application, name as you prefer, ex "SimpleCanny1"
set preferred location, uncheck "Create directory for solution" and "Add to source control", choose OK,
choose Next, uncheck "Precompiled Header" and "Security Development",
check "Empty Project" and verify "Console application" radio button is checked
choose Finish

right click in Solution Explorer, choose Add -> New Item
name C++ file as preferred, ex. "SimpleCanny1.cpp"
copy/paste the entire code from your chosen example into the .cpp file

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
(obviously if you are using a webcam example this step does not apply)

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

in VS go to:
Project -> Properties -> Configuration Properties -> VC++ Directories -> Include Directories
add the include directory: C:\OpenCV-2.4.11\opencv\build\include
(note that the **include** directory is *the same as* the precompiled binary directory used in part 1)

in VS go to:
Project -> Properties -> Configuration Properties -> VC++ Directories -> Library Directories:
add the **library** directory: C:\OpenCV-2.4.11\mybuild\lib\Debug
(note that the **library** directory is *different* from the precompiled binary directory used in part 1)

in Windows Explorer (not within Visual Studio) navigate to the **lib** directory:
C:\OpenCV-2.4.11\mybuild\lib\Debug
verify the libs listed are the same as this list:

opencv_calib3d2411d.lib
opencv_contrib2411d.lib
opencv_core2411d.lib
opencv_features2d2411d.lib
opencv_flann2411d.lib
opencv_gpu2411d.lib
opencv_haartraining_engined.lib
opencv_highgui2411d.lib
opencv_imgproc2411d.lib
opencv_legacy2411d.lib
opencv_ml2411d.lib
opencv_nonfree2411d.lib
opencv_objdetect2411d.lib
opencv_ocl2411d.lib
opencv_photo2411d.lib
opencv_stitching2411d.lib
opencv_superres2411d.lib
opencv_ts2411d.lib
opencv_video2411d.lib
opencv_videostab2411d.lib

then in VS copy/paste this list of libs into:
Project -> Properties -> Configuration Properties -> Linker -> Input -> Additional Dependencies

next in the Visual Studio toolbar, verify that "Solution Configurations" and "Solution Platforms" are set to "Debug" and "Win32", respectively

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)

now we will move on to Qt:

(Note: if you skipped the previous steps b/c you were not interested in using a full compile with Visual Studio, you still have to update your PATH variable to C:\OpenCV-2.4.11\mybuild\bin\Debug (see above))

from my MicrocontrollersAndMore GitHub page decide which example you are going to use:
CannyStillQt_Non-GUI.cpp (uses a still image, no Qt GUI)
CannyStillQt.cpp (uses a still image and a Qt GUI)
CannyWebcamQt.cpp (uses a webcam and a Qt GUI)
RedBallTrackerQt.cpp (tracks a red ball, uses a webcam and a Qt GUI)

if you are using a Qt GUI example, scroll down to "Qt GUI example instructions"
if you are using a non-GUI Qt example (i.e. CannyStillQt_Non-GUI.cpp) continue here . . .

Qt non-GUI example instructions:

Qt GUI example instructions:

start Qt, choose New Project
choose "Application" and "Qt Widgets Application", choose "Choose . . ."

on the **"Introduction and Project Location"** screen, choose Name as preferred, ex CannyStillQt1
choose "Create in:" location, ex "C:\QtProgs"
check "Use as default project location" if you plan on using the chosen location for future Qt programs
choose "Next >"

on the **"Kit Selection"** screen press "Details" to show all builds
uncheck options until only one is remaining, the Debug build for the 32-bit compile with MSVC 2013
set the "Debug" directory/name to be the combination of the directory/name from the previous screen,
for example, set "Debug" to "C:\QtProgs\QtCannyStill1"
if these are the same as the location and 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
choose "Next >"

on the **"Class Information"** screen,
verify "Base class:" is set to QMainWindow and verify "Generate form:" is checked
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
choose "Next >"

on the **"Project Management"** screen, verify "Add to version control:" is set to "<None>"
choose "Finish"

if this is your 1st time using Qt you can change things such as syntax highlighting by going to:
Tools -> Options -> Text Editor

go to your .pro file, ex "QtCannyStill1.pro", and copy/paste the following at the bottom:

```
##############################################################################
# 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) are line continuation characters

INCLUDEPATH += C:\\OpenCV-2.4.11\\opencv\\build\\include

LIBS += -LC:\\OpenCV-2.4.11\\mybuild\\lib\\Debug \
    -lopencv_calib3d2411d \
    -lopencv_contrib2411d \
    -lopencv_core2411d \
    -lopencv_features2d2411d \
    -lopencv_flann2411d \
    -lopencv_gpu2411d \
    -lopencv_haartraining_engined \
    -lopencv_highgui2411d \
    -lopencv_imgproc2411d \
    -lopencv_legacy2411d \
    -lopencv_ml2411d \
    -lopencv_nonfree2411d \
    -lopencv_objdetect2411d \
    -lopencv_ocl2411d \
    -lopencv_photo2411d \
    -lopencv_stitching2411d \
    -lopencv_superres2411d \
    -lopencv_ts2411d \
    -lopencv_video2411d \
    -lopencv_videostab2411d

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

next, 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

next, 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"
perform the same steps to remove "mainToolBar QToolBar" and "statusBar QStatusBar"
(obviously if you plan on using these for something later on do not remove them,
however for any of the example programs in this installation guide removal of these is recommended)

next add widgets to your form depending on the example you are using,
for example for "CannyStillQt.cpp", add the following widgets:

btnOpenFile
lblChosenFile
lblOriginal
lblCanny

set widget properties as shown in the video or as desired

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

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

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)