OpenCV 3.1 mit Qt 5.6.0 Installation

# Vorbedingungen

**Visual Studio 2015** (<https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx>)

**Windows 10 - 64 Bit** (dürfte wahrscheinlich auch mit < Win10 funktionieren)

# Download Vorbereitungen

**OpenCV** für Windows – Version 3.1 (<http://opencv.org/downloads.html>)

**CMake** (<https://cmake.org/download/>) –> ich habe 3.5.1. verwendet, aber ich denke, dass die neueste Version auch funktionieren wird

**Python** –> nicht sicher ob das gebraucht wird, aber ich habe „Anaconda für Windows“ in 64 Bit mit Python 3.5 installiert, weil das eine Numpy Integration besitzt, welche lt. OpenCV Tutorial gebraucht wird (bringt mehr Funktionalität in den OpenCV Build) (<https://www.continuum.io/downloads>)

**Qt 5.6.0** für Windows 64 Bit und Visual Studio 2015 (<https://www.qt.io/download-open-source/#section-2>)

**Qt5Package** – Visual Studio 2015 Extension –> zu finden in VS 2015 unter „Extras/Extensions und Updates..“, unter Online danach suchen und installieren.

# Installation

1. OpenCV 3.1 in ein Verzeichnis extrahieren, zB: „C:\OpenCV“. Darin befinden sich die Ordner „sources“ und „build“. Darin einen neuen Ordner mit „mybuild“ o.ä. erstellen, der später für den eigenen Build verwendet wird.
2. Qt 5.6.0 mit dem Installer installieren. Hierbei die einfach die Default Einstellungen verwenden. (Ich habe hierbei einige deprecated Module weggelassen und die Sources mit installiert, aber das sollte keine Rolle spielen.)
3. Umgebungsvariable „QTDIR“ für Qt erstellen, welche auf den Ordner „msvc2015\_64“ im Installationsverzeichnis von Qt zeigt. zB: „C:\Qt\Qt5.6.0\5.6\msvc2015\_64“
4. Umgebungsvariable „%QTDIR%\bin“ in die PATH Variable hinzufügen.
5. CMake ohne bestimmte Einstellungen installieren und zur PATH Variable hinzufügen lassen.
6. PC neu starten damit die Variablen aktiv werden.
7. CMake starten.
8. In „Where is the source code:“ das „source“ Verzeichnis von OpenCV setzen. zB: „C:/OpenCV /sources“
9. In „Where to build the binaries:“ das „mybuild“ Verzeichnis von OpenCV setzen. zB: „C:/OpenCV /mybuild“
10. Auf „Configure“ klicken. Hier erscheint ein Dialog, da wählt man „Visual Studio 14 2015 Win64“ aus, damit OpenCV mit 64 Bit erstellt wird. Außerdem wählen wir hier „Use default native compilers“ aus und bestätigen den Dialog.
11. Es erscheint nun eine Liste mit Einstellungsoptionen, wobei die meisten oder alle rot hinterlegt sind. Hier werden die Optionen „BUILD\_DOCS“, „BUILD\_opencv\_python2“, „BUILD\_opencv\_python3“ und „WITH\_VTK“ abgewählt. Um nun die Unterstützung für Qt zu haben, muss „WITH\_QT“ angewählt werden.
12. Nun muss einige Male auf „Configure“ gedrückt werden (ca. 3x), bis alle rot hinterlegten Einträge verschwunden sind. Hier kann es sein, dass die PATH Variable zu Qt nicht gefunden wird, was dann als rot hinterlegter Eintrag erscheint.
13. Sobald alles weiß hinterlegt ist, kann auf „Generate“ gedrückt werden, danach wird der Build erstellt. Hier kamen bei mir einige rote Einträge in der Ausgabe und eine Fehlermeldung nach dem erstellen, jedoch funktioniert z.Z. trotzdem alles!
14. CMake kann nun beendet werden.
15. Nun gehen wir in das Verzeichnis unseres Builds (C:\OpenCV\mybuild) und öffnen „OpenCV.sln“. Hier sollte der Modus auf Debug und x64 eingestellt sein, da wir für 64 Bit einen Build erstellt haben.
16. Darin das Projekt „ALL\_BUILD“ suchen, rechte Maustaste klicken und auf Erstellen klicken. (Damit werden die \*.lib und \*.dll Daten für das Debugging erstellt) Hierbei sollten bei dem Erstellen keine Fehler auftreten. Danach wird der Modus auf Release geändert und für den Release Modus das Projekt „ALL\_BUILD“ nochmals erstellt. Hierbei sollten wieder keine Fehler auftreten.
17. Hat alles geklappt, klickt man mit der rechten Maustaste auf „INSTALL“ und Erstellen, was unter „CMakeTargets“ zu finden ist. Dies macht man wieder für Debug und Release extra. Mithilfe diesen Schrittes, werden alle \*.lib und \*.dll Daten in einen eigenen Ordner verschoben, welcher unter „..\mybuild\install\x64\vc14\..“ zu finden ist.
18. Den Ordner „install“ kopiert man nun in das OpenCV Verzeichnis (C:\OpenCV).
19. Um OpenCV einheitlich verwenden zu können wird eine Umgebungsvariable „OPENCV\_DIR“ angelegt, welche auf den Ordner „install\x64\vc14“ aus dem OpenCV Verzeichnis zeigt.
20. Des Weiteren muss in der PATH Variable ein Eintrag mit „%OPENCV\_DIR%\bin“ ergänzt werden.
21. PC neu starten damit die Variablen aktiv werden.
22. OPTIONAL: In Visual Studio muss eventuell noch die Qt Version mit dem zugehörigen Pfad eingetragen werden, damit die Qt Funktionen verfügbar sind. Dazu öffnet man in Visual Studio den Menu Punkt „QT5/QT Options“. Falls hier schon die aktuellste Version eingetragen ist, muss man nichts mehr machen. Ansonsten klickt man auf „Add“ und schreibt den Versionsnamen und fügt den Pfad für den „mcsv2015\_64“ Ordner ein (z.B: C:\Qt\Qt5.6.0\5.6\msvc2015\_64).

FERTIG! ☺

Das Projekt sollte damit funktionieren und sich ohne Probleme ausführen lassen. Mit dem eigenen Build von OpenCV haben sich die gelinkten Bibliotheken geändert und die damit in Verbindung stehenden Eigenschaftsblätter von OpenCV. Für Qt wird hierbei kein eigenes Eigenschaftsblatt gebraucht, da mit der Erstellung eines Qt Projektes über die Visual Studio Extension, die Abhängigkeiten automatisch mit der Umgebungsvariable „QTDIR“ eingebunden werden.