Installation von OpenCV 3 auf Raspberry Pi 3

Voraussetzungen :

-Raspberry Pi 3 mit vollständiger Raspbian-Installation

-Zugriff entweder lokal, über SSH oder über Remotedesktop (*xrdp*)

-Ausreichend Platz auf der MicroSD-Karte (min. 8GB FREI)

*Achtung: Die Installation von OpenCV auf Raspbian beinhaltet* ***zwingend*** *die Kompilierung der gesamten Bibliotheken. Die reine Kompilationszeit beträgt ca. zwei bis zweieinhalb Stunden.*

Vor der Installation ist wie üblich ein update/upgrade durchzuführen:

$ sudo aptitude update

$ sudo aptitude upgrade

Außerdem wird noch CMake benötigt, um den Buildprozess zu konfigurieren.

$ sudo apt-get install build-essential cmake pkg-config

Damit der Raspberry Bilder und Videos von der Platte laden kann, müssen noch folgende Dependencies installiert werden:

$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev

$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

$ sudo apt-get install libxvidcore-dev libx264-dev

Damit OpenCV GUIs erzeugen kann, muss die GTK-Entwicklungsbibliothek installiert werden:

$ sudo apt-get install libgtk2.0-dev

Um Matrixoperationen in OpenCV zu beschleunigen, sollten weitere Pakete installiert werden:

$ sudo apt-get install libatlas-base-dev gfortran

Um sicherzustellen, dass für die Kompilierung Python-headerfiles vorhanden sind, müssen dev-pakete sowohl von Python2 und Python3 installiert warden:

$ sudo apt-get install python2.7-dev python3-dev

Wenn jetzt alle obigen Voraussetzungen erfüllt sind, kann der Source Code von OpenCV beschafft werden. Die Versionsnummer ist hier durch die jeweils aktuelle Version zu ersetzen. Heruntergeladen wird er ins aktuelle home-directory.

$ cd ~

$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.2.0.zip

$ unzip opencv.zip

$ rm opencv.zip

Um auf Features wie SIFT und SURF (Pattern matching, Objekterkennung) zugreifen zu können, müssen noch zusätzliche Pakete beschafft werden:

$ wget -O opencv\_contrib.zip <https://github.com/Itseez/opencv_contrib/archive/3.2.0.zip>

$ unzip opencv\_contrib.zip

$ rm opencv\_contrib.zip

Für die Installation der Python-Erweiterungen muss außerdem pip, der Python Package Manager installiert werden:

$ wget https://bootstrap.pypa.io/get-pip.py

$ sudo python get-pip.py

Damit sich die Packages die wir installieren, und auch unsere Python-Konfiguration später nicht mit anderen Projekten beißen, muss ein Python Virtual Environment erstellt werden:

$ sudo pip install virtualenv virtualenvwrapper

$ sudo rm -rf ~/.cache/pip

Dann müssen unserer “.profile”-Datei in unserem Home-Verzeichnis ein paar Zeilen hinzugefügt werden, um einen einfachen Zugriff auf unsere virtuelle Konfiguration zu erlauben.

$ echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile

$ echo "export WORKON\_HOME=$HOME/.virtualenvs" >> ~/.profile

$ echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile

Um das Profil nun neu zu laden sollte der folgende Befehl ausgeführt werden:

$ source ~/.profile

Dieser Befehl sollte außerdem jedes Mal ausgeführt werden, wenn mit dem virtual Environment gearbeitet werden soll.

Als nächstes muss dementsprechend ein virtual Environment eingerichtet werden, welches wir für unser Projekt benutzen können (Unter der Annahme, dass wir Python3 nutzen):

$ mkvirtualenv cv -p python3

Unser neues virtual Environment hat also den Namen „cv“. Alle packages, die wir dort installieren, werden nicht für unsere globale Python-Installation verfügbar sein, wie auch unsere virtuell installierten Packages nicht global verfügbar sein werden.

Um das virtual Environment nutzen zu können, muss der „workon“-Befehl ausgeführt werden. Es empfiehlt sich, vor jedem Start die „.profile“-Datei neu zu laden:

$ source ~/.profile

$ workon cv

Die Bestätigung, dass wir uns in unserem neuen virtual Environment befinden, liefert das (cv)-Präfix an unserer Terminal-Zeile.

Sofern wir uns jetzt noch in unserem virtual Environment befinden, können wir die packages installieren, die wir für unsere OpenCV-Installation benötigen. Zuerst NumPy, welches numerische Berechnungen erheblich vereinfacht. Die Installation dauert recht lange (~10 Min) und manchmal scheint nichts zu passieren, also einfach abwarten:

$ pip install numpy

Es empfiehlt sich, vor dem kompilieren nochmals sicherzustellen, dass wir uns noch in „cv“ befinden.

Die Kompilierung muss jetzt zunächst einmal konfiguriert werden:

$ cd ~/opencv-3.1.0/

$ mkdir build

$ cd build

$ cmake -D CMAKE\_BUILD\_TYPE=RELEASE \ -D CMAKE\_INSTALL\_PREFIX=/usr/local \ -D INSTALL\_PYTHON\_EXAMPLES=ON \ -D OPENCV\_EXTRA\_MODULES\_PATH=~/opencv\_contrib-3.2.0/modules \ -D BUILD\_EXAMPLES=ON ..

Im Output sollte jetzt sichergestelt warden, dass der “Python 3”-Abschnitt auf unsere Python-Installation innerhalb des virtual Environments zeigt, gleich für NumPy.

Wir sind nun bereit, OpenCV zu kompilieren. Dieser Teil wird mindestens zwei Stunden in Anspruch nehmen:

$ make –j4

Das „-j4“ sorgt dafür, dass für die Kompilierung alle vier Prozessorkerne verwendet werden.

Wenn die Kompilierung beendet wurde, muss OpenCV noch installiert werden:

$ sudo make install

$ sudo ldconfig

Nun sollten wir sicherstellen, dass sich das kompilierte package im „site-packages“-Verzeichnis befindet.

$ cd usr/local/lib/python3.5/site-packages/

$ ls

Es sollte eine Datei „cv.cpython35m.so“ oder „cv2.so“ vorhanden sein.

Im ersteren Fall muss die erstellte Datei noch in “cv2.so” umbenannt werden:

$ sudo mv cv2.cpython-34m.so cv2.so

Die OpenCV-Bindings müssen jetzt noch auf unser virtual Environment übertragen werden:

$ cd ~/.virtualenvs/cv/lib/python3.4/site-packages/

$ ln -s /usr/local/lib/python3.4/site-packages/cv2.so cv2.so

Die Installation ist jetzt vollständig. Wir sollten nun überprüfen ob alles soweit funktioniert hat.

Dafür sollte ein neues Terminal geöffnet werden, außerdem müssen wir wieder unser virtual Environment aktivieren:

$ source ~/.profile

$ workon cv

Wir können jetzt python starten, die package importieren und die Versionsnummer abfragen:

$ python

>>> import cv2

>>> cv2.\_\_version\_\_

'3.2.0'

>>> exit()

Sollte unser Output so aussehen, haben wir OpenCV erfolgreich installiert und können unsere Quelldateien entfernen:

$ cd ~

$ rm -rf opencv-3.1.0 opencv\_contrib-3.1.0