OpenCV 3.x Installation for Ubuntu 16.06

OpenCV is an open source library for image processing thats supported in multiple languages. This guide explains installation steps for **Python** only.

Prerequisites:

1.1 Python : Install python from miniconda. Avoid anaconda since it comes with too many redundant packages.

1.1.1 Steps to install python:
1. Download the required python from the above link. (Python 3.6 Recommended)
2. In terminal run:

```
$ bash <yourpythonfile.sh>
```

Note: Everytime it asks about updating path and where to install leave it at its default. (Just press enter.)

1.2 Changing the system python (in order to use sudo with miniconda python):

1. First add all the pythons in a list like this:

   ```
   $ sudo update-alternatives --install /usr/bin/python python /usr/bin/pyt
   hon2.7 1
   $ sudo update-alternatives --install /usr/bin/python python /usr/bin/pyt
   hon3.5 2
   $ sudo update-alternatives --install /usr/bin/python python /home/rakshi
   th/miniconda3/bin/python3.6 3
   ```

   (Enter your corresponding miniconda python path, the last number in each command is basically a priority)

2. Everytime you want to give sudo permission to minconda python you can do so by:

   ```
   $ sudo update-alternatives --config python
   ```

3. Then type the corresponding index number of the python you want to give sudo permission to and press enter.

4. Make sure to change the permission back to system python2.7 once your done. Since Ubuntu needs this for installing system packages.

1.3 Assigning root permissions to conda environment: [Warning: Be very careful while doing this. One mistake can destroy your system.]
Conda sometimes throws permission error in order to fix this. Run this in terminal:

```
$ sudo chown -R USERNAME /home/USERNAME/miniconda3
```

Here replace **USERNAME** with your username and the path with the path of your miniconda.

BE VERY CAREFULL WITH THE PATH. YOU WILL HAVE TO REINSTALL OS IF SOMETHING GOES WRONG.

1.4 Installing OpenCV prerequisite packages:
Make sure invoking `$ sudo python` shows system python 2.7, if not then follow step **1.2** to change it to system python 2.7.

Update Packages:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Install OS Packages:

```
$ sudo apt-get install build-essential checkinstall cmake pkg-config yasm
$ sudo apt-get install git gfortran
$ sudo apt-get install libjpeg8-dev libjasper-dev libpng12-dev
$ sudo apt-get install libtiff5-dev
$ sudo apt-get install libavcodec-dev libavutil-dev libavfilter-dev libavfor
mat-dev libavresample-dev ffmpeg
$ sudo apt-get install libswscale-dev libdc1394-22-dev
$ sudo apt-get install libxine2-dev libv4l-dev
$ sudo apt-get install libgstreamer0.10-dev libgstreamer-plugins-base0.10-de
v
$ sudo apt-get install qt5-default libgtk2.0-dev libtbb-dev
$ sudo apt-get install libopenblas-dev libatlas-base-dev liblapack-dev libei
gen3-dev
$ sudo apt-get install libfaac-dev libmp3lame-dev libtheora-dev
$ sudo apt-get install libvorbis-dev libxvidcore-dev
$ sudo apt-get install sphinx-common libtbb-dev
$ sudo apt-get install libopencore-amrnb-dev libopencore-amrwb-dev libopenex
r-dev
$ sudo apt-get -qq install libopencv-dev libqt4-dev
$ sudo apt-get install x264 v4l-utils
$ sudo apt-get install libprotobuf-dev protobuf-compiler
$ sudo apt-get install libgoogle-glog-dev libgflags-dev
$ sudo apt-get install libgphoto2-dev libhdf5-dev doxygen
```

1.5 Installing Nvidia Cuda Packages for GPU support: [Optional]
The official NVIDIA Guide is very messy and complicated. These steps are taken from there and are arranged in sequence:

Firstly Check if your GPU is supported for cuda library in this link.

Second if you want to follow every step in NVIDIA's documentation for cuda toolkit here is the link.

This is the official link for cuDNN. You need to sign up as Nvidia Developer to download cudnn and its manual to install. (Its free.)

Simplified steps:

1.5.1 Prerequisite steps:
i. Check if you have a CUDA-Capable GPU. (its better to do this even if your GPU is listed in that website.):

```
$ lspci | grep -i nvidia
```

If you do not see any settings, update the PCI hardware database that Linux maintains by entering `update-pciids` (generally found in `/sbin`) at the command line and rerun the previous `lspci` command.
ii. Verify you have a supported version of linux:

```
$ uname -m && cat /etc/*release
```

iii. Check if you have the correct version of gcc:

```
$ gcc --version
```

iv. Verify the system has the correct kernel headers and development packages installed:

```
$ uname -r
```

v. Then run:

```
$ sudo apt-get install linux-headers-$(uname -r)
```

1.5.2 Install cuda sdk:
Download the correct version in this link. Make sure you are downloading `deb (local)` . The file will be about 1.2 GB in size.

Installation:
i. Change the file name to the downloaded version:

```
 $ sudo dpkg -i <yourdownloadedfilename>.deb
```

ii. `$ sudo apt-key add /var/cuda-repo-<version>/7fa2af80.pub`
iii. `$ sudo apt-get update`
iv. `$ sudo apt-get install cuda`
v. `$ sudo apt-get install cuda-drivers`
vi. `$ /usr/bin/nvidia-persistenced --verbose`
vii. Optional Step:

```
sudo apt-get install g++ freeglut3-dev build-essential libx11-dev \ libxmu-d
ev libxi-dev libglu1-mesa libglu1-mesa-dev
```

Finally update your system `PATH` like this:

```
$ sudo nano /etc/environment
```

Add this: `/usr/local/cuda-<yourversion>/bin` towards the end (replace < yourversion > with the one you installed, provided you installed cuda at its default path. So it would look something like this:

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/game
s:/usr/local/games:/usr/local/cuda-9.1/bin"
```

`ctrl+O` and then `ctrl+X`

Reboot the system and check if everything was installed properly. To check run:

```
$ nvidia-smi
```

This should display your graphic card with all its details and cuda driver version.
Also check:

```
$ nvcc --version
```

If this fails to give any output even after rebooting. Reinstall cuda toolkit:

```
$ sudo apt-get install cuda-toolkit
```

Check with `nvcc` command again once this is done. It should be fixed.

1.5.3 Installing cuDNN:
The Official Documentation is good for this just follow it. You need to sign up with NVIDIA as a developer in order to install this. You can download the 3 files here.

Just download the `.deb` files and follow step `2.3.2` in this Installation Guide.

Finally run in terminal:

```
$ sudo apt-get install libcupti-dev
```

Reboot once everything is installed.

1.6 Download OpenCV:
1. Create an empty folder in **Documents** and name it **OpenCV**.
2. Open **Terminal** and navigate to this folder.
`$ cd Documents/OpenCV`
3. Download OpenCV here by typing (replace 3.4.0 with latest) :
`$ wget https://github.com/Itseez/opencv/archive/3.4.0.zip`
4. `$ unzip 3*.zip`
5. **[Optional]** Download OpenCV_Contrib by typing:
`$ wget https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip`
and unzip this too.

1.7 Install Intel MKL [Optional]:

1. Download free MKL.
2. Extract to `/tmp` and type:

```
./install_GUI.sh
```

1. Choose GUI option “root as sudo”
2. This will install to `/opt/intel`

Installing OpenCV with Cmake

1. Switch to the python version you want to install OpenCV for by using **Step 1.2** and install these `pip` packages (only `numpy` is required, rest are optional and handy):

   ```
   pip install numpy scipy matplotlib scikit-image scikit-learn ipython
   ```

2. Navigate to **OpenCV** folder in **Documents**.

3. Extract the files which were downloaded in **Step 1.6**.

4. Navigate to the extracted **opencv-3.4.0** folder and create a new folder called **build**.

5. Navigate to this newly created **build** folder and open Terminal here.

6. Run the following CMake Command:
   Note: If you wish to install for python2 then change the flags and paths accordingly.
   For **CPU Only** with `opencv_contrib` : (Replace all the paths with your paths as needed)

   ```
   cmake -D CMAKE_BUILD_TYPE=Release -D CMAKE_INSTALL_PREFIX=/usr/local -D
   OPENCV_EXTRA_MODULES_PATH=/home/rakshith/OpenCV/opencv_contrib-3.4.0/mod
   ules -D WITH_CUBLAS=ON -D WITH_TBB=ON -D WITH_V4L=ON -D WITH_QT=ON -D WI
   TH_OPENGL=ON -D BUILD_PERF_TESTS=OFF -D BUILD_TESTS=OFF -D BUILD_opencv_
   python2=OFF -D BUILD_opencv_python3=ON -D PYTHON_DEFAULT_EXECUTABLE=/hom
   e/rakshith/miniconda3/bin/python3.6 -D PYTHON3_INCLUDE_DIR=/home/rakshit
   h/miniconda3/include/python3.6m -D PYTHON3_LIBRARY=/home/rakshith/minico
   nda3/lib/libpython3.6m.so -D PYTHON3_PACKAGES_PATH=/home/rakshith/minico
   nda3/lib/python3.6/site-packages ..
   ```

   For **GPU+CPU** with `opencv_contrib` :(Replace all the paths with your paths as needed)

   ```
   cmake -D CMAKE_BUILD_TYPE=Release -D CMAKE_INSTALL_PREFIX=/usr/local -D
   ```

```
OPENCV_EXTRA_MODULES_PATH=/home/rakshith/OpenCV/opencv_contrib-3.4.0/mod
ules -D WITH_CUDA=ON -D CUDA_GENERATION=Auto -D WITH_CUBLAS=ON -D WITH_T
BB=ON -D WITH_V4L=ON -D WITH_QT=ON -D WITH_OPENGL=ON -D BUILD_PERF_TESTS
=OFF -D BUILD_TESTS=OFF -DCUDA_NVCC_FLAGS="-D_FORCE_INLINES" -D ENABLE_F
AST_MATH=1 -D CUDA_FAST_MATH=1 -D BUILD_opencv_python2=OFF -D BUILD_open
cv_python3=ON -D PYTHON_DEFAULT_EXECUTABLE=/home/rakshith/miniconda3/bin
/python3.6 -D PYTHON3_INCLUDE_DIR=/home/rakshith/miniconda3/include/pyth
on3.6m -D PYTHON3_LIBRARY=/home/rakshith/miniconda3/lib/libpython3.6m.so
 -D PYTHON3_PACKAGES_PATH=/home/rakshith/miniconda3/lib/python3.6/site-p
ackages ..
```

7. `make -j4`

8. `sudo make install`

Thats it, we can now use OpenCV with python.

Notes

Some OS Packages may not work with higher versions of Ubuntu like `libjasper-dev`, install equivalent packages.

Debug

Get OpenCV build information:

```
$ python
import cv2
print(cv2.getBuildInformation())
```

To uninstall OpenCV:
1. Navigate to the **build** directory.
2. `$ sudo make uninstall`
3. Delete **OpenCV** directory.