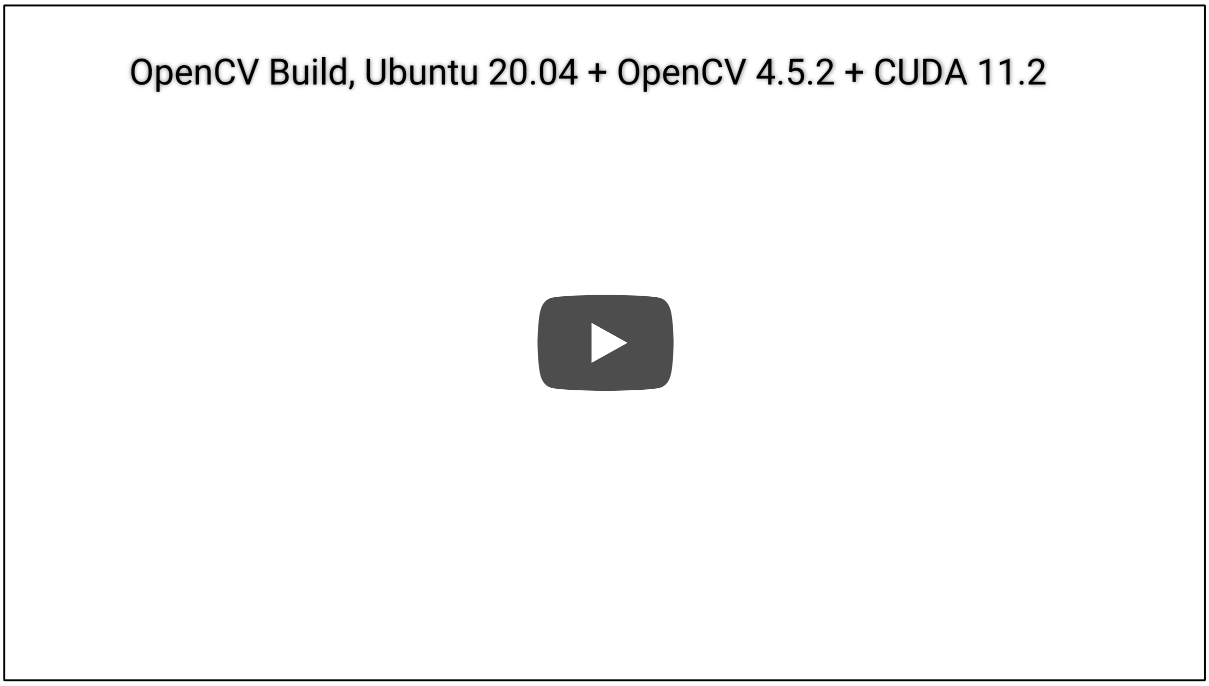
0. OpenCV Build, Ubuntu 20.04 + OpenCV 4.5.2 + CUDA 11.2

Tags	build cuda ubuntu
Last edited	July 22, 2021 9:21 PM
Keyword	Empty

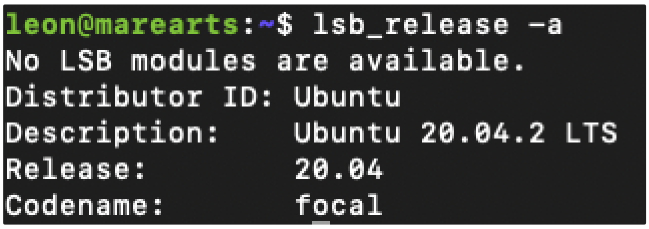


OpenCV Build, Ubuntu 20.04 + OpenCV 4.5.2 + CUDA 11.2

0. Check your system environment

- **Check your Ubuntu version**

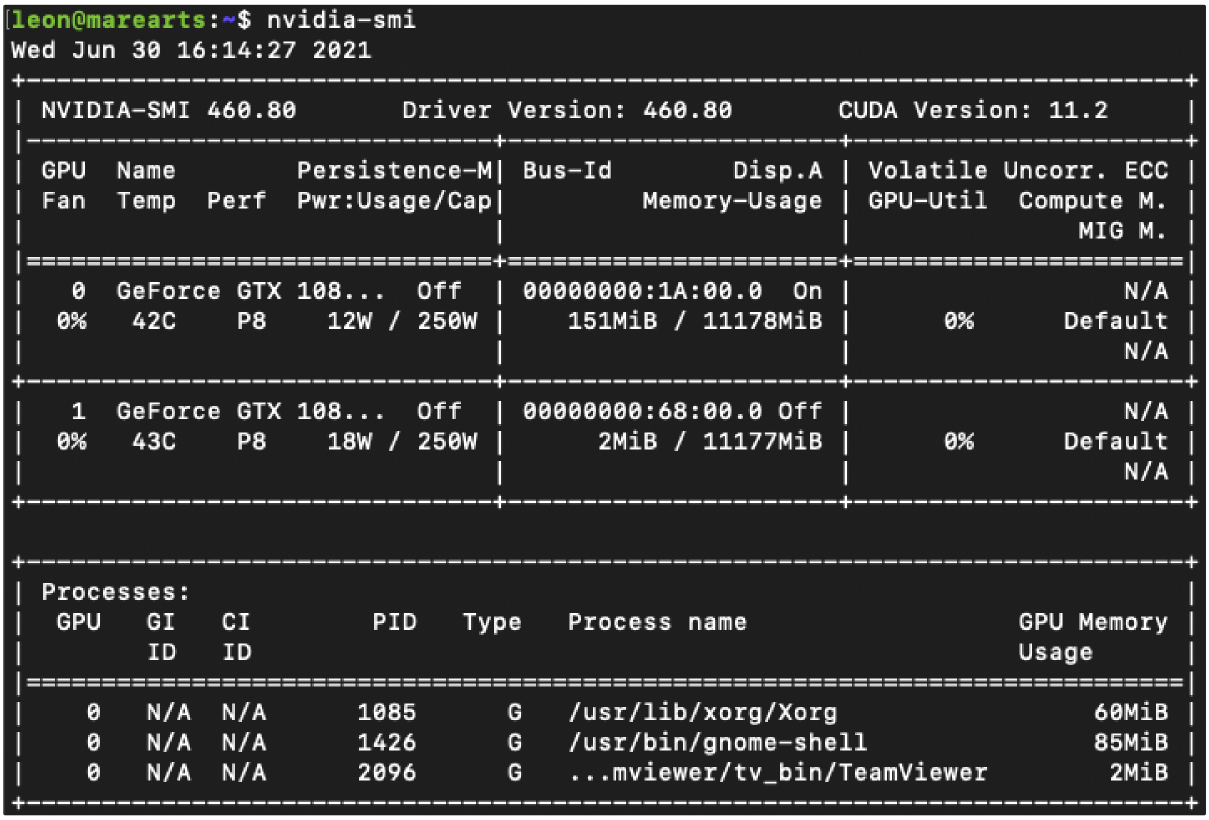
```
> lsb_release -a
```



```
leon@marearts:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.2 LTS
Release:        20.04
Codename:       focal
```

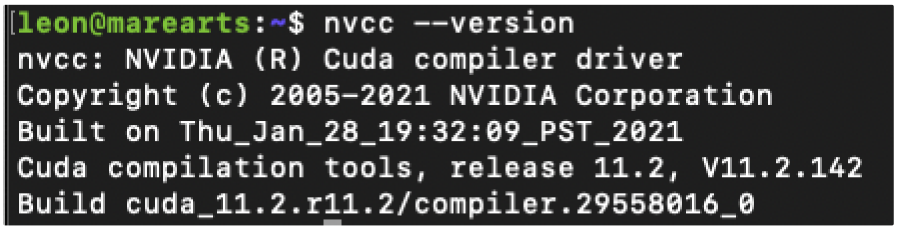
- **Check your CUDA, GPU driver is installed properly.**

```
> nvidia-smi > nvcc --version
```



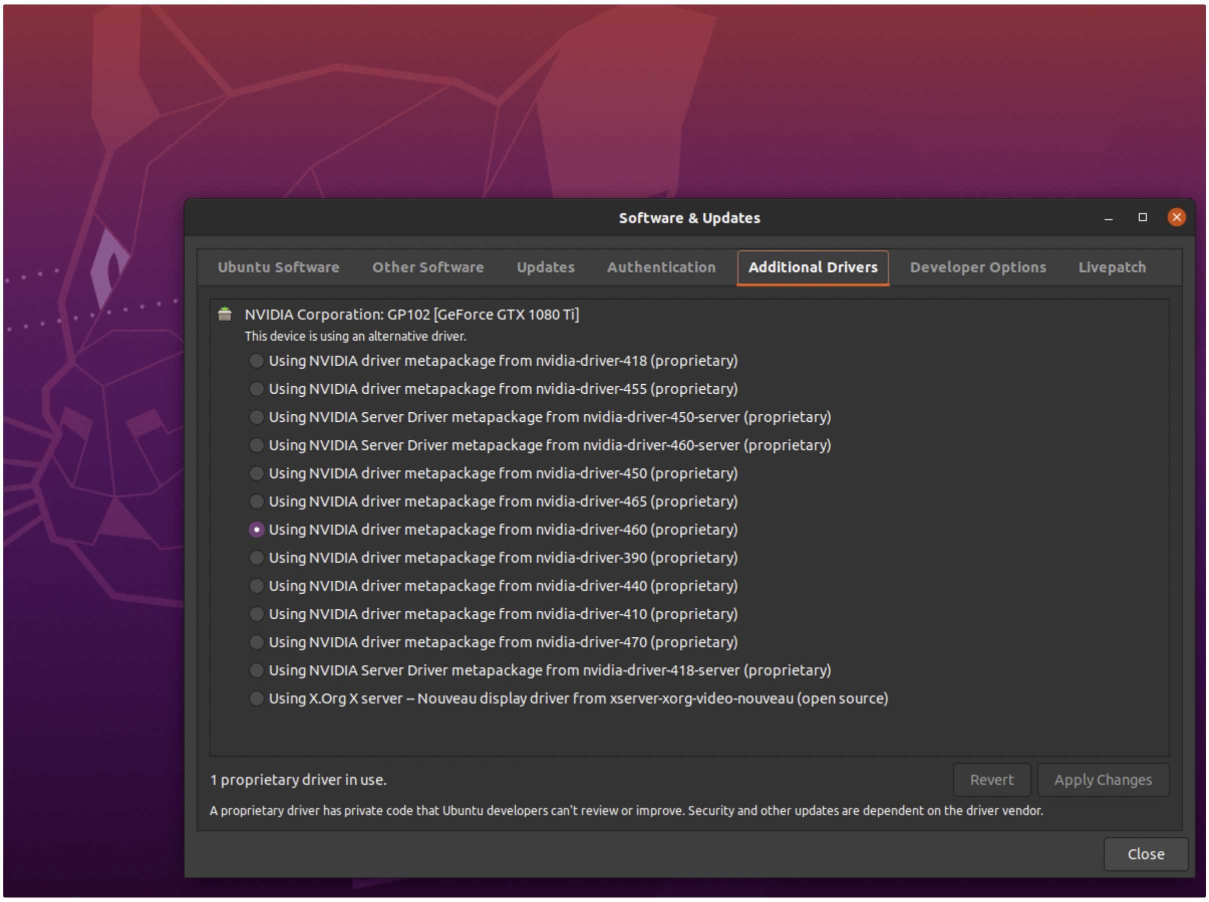
```
leon@marearts:~$ nvidia-smi
Wed Jun 30 16:14:27 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.80       Driver Version: 460.80       CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GeForce GTX 108...  Off  | 00000000:1A:00.0  On |                  N/A |
|  0%   42C    P8    12W / 250W |    151MiB / 11178MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 108...  Off  | 00000000:68:00.0 Off |                  N/A |
|  0%   43C    P8    18W / 250W |      2MiB / 11177MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      1085      G   /usr/lib/xorg/Xorg                 60MiB |
|    0   N/A  N/A      1426      G   /usr/bin/gnome-shell               85MiB |
|    0   N/A  N/A      2096      G   ...mviewer/tv_bin/TeamViewer        2MiB |
+-----------------------------------------------------------------------------+
```



```
leon@marearts:~$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2021 NVIDIA Corporation
Built on Thu_Jan_28_19:32:09_PST_2021
Cuda compilation tools, release 11.2, V11.2.142
Build cuda_11.2.r11.2/compiler.29558016_0
```

- **You also can check driver version on Software & Updates tool in Window.**
    - In my case, driver version is 460.80



NVIDIA Corporation: GP102 [GeForce GTX 1080 Ti]
This device is using an alternative driver.

○ Using NVIDIA driver metapackage from nvidia-driver-418 (proprietary)
○ Using NVIDIA driver metapackage from nvidia-driver-455 (proprietary)
○ Using NVIDIA Server Driver metapackage from nvidia-driver-450-server (proprietary)
○ Using NVIDIA Server Driver metapackage from nvidia-driver-460-server (proprietary)
○ Using NVIDIA driver metapackage from nvidia-driver-450 (proprietary)
○ Using NVIDIA driver metapackage from nvidia-driver-465 (proprietary)
● Using NVIDIA driver metapackage from nvidia-driver-460 (proprietary)
○ Using NVIDIA driver metapackage from nvidia-driver-390 (proprietary)
○ Using NVIDIA driver metapackage from nvidia-driver-440 (proprietary)
○ Using NVIDIA driver metapackage from nvidia-driver-410 (proprietary)
○ Using NVIDIA driver metapackage from nvidia-driver-470 (proprietary)
○ Using NVIDIA Server Driver metapackage from nvidia-driver-418-server (proprietary)
○ Using X.Org X server – Nouveau display driver from xserver-xorg-video-nouveau (open source)

1 proprietary driver in use.

A proprietary driver has private code that Ubuntu developers can't review or improve. Security and other updates are dependent on the driver vendor.

Reference how to install CUDA on ubuntu:

- link1 : Installing CUDA 10.1 on Ubuntu 20.04
- link2 : Installing Tensorflow with CUDA & cuDNN GPU support on Ubuntu 20.04 and charge through your Linear Algebra calculations

1. Preparation

- **Install update and upgrade your system**

```
> sudo apt update > sudo apt upgrade
```

- **Install required libraries**

```
# Generic tools: > sudo apt install build-essential cmake pkg-
config unzip yasm git checkinstall # Image I/O libs > sudo apt
install libjpeg-dev libpng-dev libtiff-dev # Video/Audio Libs -
FFMPEG, GSTREAMER, x264 ans so on > sudo apt install libavcodec-dev
libavformat-dev libswscale-dev libavresample-dev > sudo apt install
libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev > sudo apt
install libxvidcore-dev x264 libx264-dev libfaac-dev libmp3lame-dev
libtheora-dev > sudo apt install libfaac-dev libmp3lame-dev
libvorbis-dev # OpenCore - Adaptive Multi Rate Narrow Band (AMRNB)
and Wide Band (AMRWB) speech codec > sudo apt install libopencore-
amrnb-dev libopencore-amrwb-dev # Cameras programming interface
libs > sudo apt-get install libdc1394-22 libdc1394-22-dev libxine2-
dev libv4l-dev v4l-utils > cd /usr/include/linux > sudo ln -s -f
../libv4l1-videodev.h videodev.h > cd ~ # GTK lib for the graphical
user functionalites coming from OpenCV highghui module > sudo apt-
get install libgtk-3-dev # Python libraries for python3: > sudo
apt-get install python3-dev python3-pip > sudo -H pip3 install -U
pip numpy > sudo apt install python3-testresources # Parallelism
library C++ for CPU > sudo apt-get install libtbb-dev #
Optimization libraries for OpenCV > sudo apt-get install libatlas-
base-dev gfortran # Optional libraries: > sudo apt-get install
libprotobuf-dev protobuf-compiler > sudo apt-get install libgoogle-
glog-dev libgflags-dev > sudo apt-get install libgphoto2-dev
libeigen3-dev libhdf5-dev doxygen
```

2. OpenCV Build

- **Download**

```
> cd ~/Downloads > wget -O opencv.zip
https://github.com/opencv/opencv/archive/refs/tags/4.5.2.zip > wget
-O opencv_contrib.zip
https://github.com/opencv/opencv_contrib/archive/refs/tags/4.5.2.zip
> unzip opencv.zip > unzip opencv_contrib.zip
```

▶ **Set virtualenv environment for python (Optional)**

- **go to installation folder**

```
> cd opencv-4.5.2 > mkdir build > cd build
```

- **make source code**

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \ -D
CMAKE_INSTALL_PREFIX=/usr/local \ -D WITH_TBB=ON \ -D
ENABLE_FAST_MATH=1 \ -D CUDA_FAST_MATH=1 \ -D WITH_CUBLAS=1 \ -D
WITH_CUDA=ON \ -D BUILD_opencv_cudacodec=OFF \ -D WITH_CUDNN=ON \ -
D OPENCV_DNN_CUDA=ON \ -D CUDA_ARCH_BIN=7.5 \ -D WITH_V4L=ON \ -D
WITH_QT=OFF \ -D WITH_OPENGL=ON \ -D WITH_GSTREAMER=ON \ -D
OPENCV_GENERATE_PKGCONFIG=ON \ -D OPENCV_PC_FILE_NAME=opencv.pc \ -
D OPENCV_ENABLE_NONFREE=ON \ -D
OPENCV_PYTHON3_INSTALL_PATH=~/.virtualenvs/cv/lib/python3.8/site-
packages \ -D PYTHON_EXECUTABLE=~/.virtualenvs/cv/bin/python \ -D
OPENCV_EXTRA_MODULES_PATH=~/Downloads/opencv_contrib-4.5.2/modules
\ -D INSTALL_PYTHON_EXAMPLES=OFF \ -D INSTALL_C_EXAMPLES=OFF \ -D
BUILD_EXAMPLES=OFF ..
```

- **other options**

```
-D CUDA_ARCH_BIN:STRING=6.0 6.1 7.0 7.5 -D BUILD_opencv_world=ON -D
BUILD_opencv_contrib_world=ON -D
CUDA_TOOLKIT_ROOT_DIR=/usr/local/cuda-11.4
```

▶ **Refer to My Configuration Report**

- **Build opencv library**

```
# Check how many cpu cores > nproc 16 # build > make -j16 # install
all built libs into your system > sudo make install
```

- include the libs in your environment

```
> sudo /bin/bash -c 'echo "/usr/local/lib" >>
/etc/ld.so.conf.d/opencv.conf' > sudo ldconfig
```

3. Test

Example c++ source code

- 👉 Note change filename properly!

▶ Get example image

```
#include <iostream> #include <ctime> #include <cmath> #include
"bits/time.h" #include <opencv2/core.hpp> #include
<opencv2/highgui.hpp> #include <opencv2/imgproc.hpp> #include
<opencv2/imgcodecs.hpp> #include <opencv2/core/cuda.hpp> #include
<opencv2/cudaarithm.hpp> #include <opencv2/cudaimgproc.hpp> #define
TestCUDA true int main() { std::clock_t begin = std::clock(); try {
cv::String filename = "./example.jpg"; //change file name properly
cv::Mat srcHost = cv::imread(filename, cv::IMREAD_GRAYSCALE);
for(int i=0; i<1000; i++) { if(TestCUDA) { cv::cuda::GpuMat dst,
src; src.upload(srcHost);
//cv::cuda::threshold(src,dst,128.0,255.0, CV_THRESH_BINARY);
cv::cuda::bilateralFilter(src,dst,3,1,1); cv::Mat resultHost;
dst.download(resultHost); } else { cv::Mat dst;
cv::bilateralFilter(srcHost,dst,3,1,1); } }
//cv::imshow("Result",resultHost); //cv::waitKey(); } catch(const
cv::Exception& ex) { std::cout << "Error: " << ex.what() <<
std::endl; } std::clock_t end = std::clock(); std::cout <<
double(end-begin) / CLOCKS_PER_SEC << std::endl; }
```

Compile code & execute

```
> g++ test.cpp `pkg-config opencv --cflags --libs` -o test > ./test
```

Troubleshoot

🤔 libcudnn symbolic link error

```
/sbin/ldconfig.real: /usr/local/cuda-11.2/targets/x86_64-
linux/lib/libcudnn.so.8 is not a symbolic link
```

/sbin/ldconfig.real: /usr/local/cuda-11.2/targets/x86_6…

Problems ------- > sudo ldconfig /sbin/ldconfig.real: /usr/local/cuda-11.2/targets/x86_64-

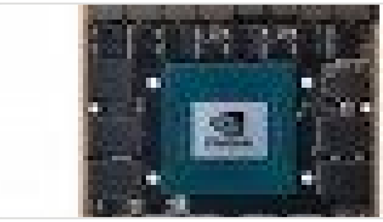
https://study.marearts.com/2021/06/sbinldconfigreal-usrlo…

🤔 CUDA_ARCH_BIN, compute capability value table

CUDA_ARCH_BIN Table for gpu type

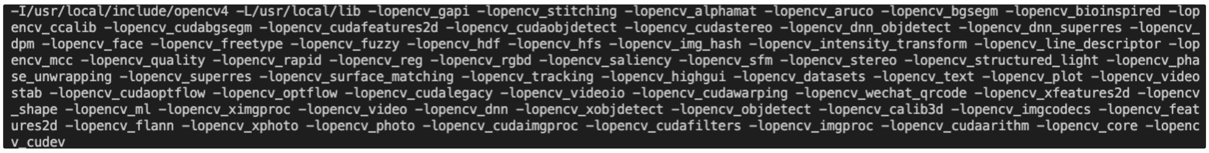
CUDA_ARCH_BIN Table for gpu type Jetson Products GPU Compute Capability Jetson AGX Xavier 7.2 Jetson Nano 5.3

https://study.marearts.com/2021/06/cudaarchbin-table-for-…



🤔 Test if your system can find the OpenCV paths, run the following command

```
> pkg-config --cflags --libs opencv
```



```
-I/usr/local/include/opencv4 -L/usr/local/lib -lopencv_gapi -lopencv_stitching -lopencv_alphamat -lopencv_aruco -lopencv_bgsegm -lopencv_bioinspired -lop
encv_ccalib -lopencv_cudabgsegm -lopencv_cudafeatures2d -lopencv_cudaobjdetect -lopencv_cudastereo -lopencv_dnn_objdetect -lopencv_dnn_superres -lopencv_
dpm -lopencv_face -lopencv_freetype -lopencv_fuzzy -lopencv_hdf -lopencv_hfs -lopencv_img_hash -lopencv_intensity_transform -lopencv_line_descriptor -lop
encv_mcc -lopencv_quality -lopencv_rapid -lopencv_reg -lopencv_rgbd -lopencv_saliency -lopencv_sfm -lopencv_stereo -lopencv_structured_light -lopencv_pha
se_unwrapping -lopencv_superres -lopencv_surface_matching -lopencv_tracking -lopencv_highgui -lopencv_datasets -lopencv_text -lopencv_plot -lopencv_video
stab -lopencv_cudaoptflow -lopencv_optflow -lopencv_cudalegacy -lopencv_videoio -lopencv_cudawarping -lopencv_wechat_qrcode -lopencv_xfeatures2d -lopencv
_shape -lopencv_ml -lopencv_ximgproc -lopencv_video -lopencv_dnn -lopencv_xobjdetect -lopencv_objdetect -lopencv_calib3d -lopencv_imgcodecs -lopencv_feat
ures2d -lopencv_flann -lopencv_xphoto -lopencv_photo -lopencv_cudaimgproc -lopencv_cudafilters -lopencv_imgproc -lopencv_cudaarithm -lopencv_core -lopenc
v_cudev
```

- If it gives you an error message, your PKG_CONFIG_PATH environment variable may not be pointing to OpenCV and you must add the path on it. Try to run this and the test once again

```
> export
PKG_CONFIG_PATH="$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig"
```

- You can append the last command on a profile script to avoid the need to run it every time, try .bashrc:

```
> vi ~/.bashrc
```

🙇 Thank you!

🎁 Source code, Material(pdf) and example images 👇

OpenCV Lecture

OpenCV Lecture class materialsPPT, source code and images

https://www.buymeacoffee.com/tRUrg28/e/18943

hits 5 / 2661