When it comes to embedded computer vision, fractions of code acceleration are regarded as a huge success for programmers. Today, I’ll explain how to build a customized OpenCV for Raspberry Pi as one of the most famous single-board computers. By following these simple tips, you’ll experience a 2-3x faster OpenCV on your board.

Raspberry Pi 2 and higher versions have multi-core CPUs that support ARM NEON technology. Clearly, a code that benefits from these two options will run much faster than a simple bare code. The good news is that most OpenCV functions are parallelized on CPU and a limited number of them benefit from NEON C intrinsics. You can check this by digging into some of the source codes (e.g. KLT tracker implementation in opencv/src/modules/video/src/lkpyramid.cpp) and look for parallel\_for\_and if CV\_NEON statements.

The bad news is that, if you have previously built OpenCV on your board, it is most likely that your library doesn’t benefit from these options. You can simply check by running the following script:

**#include**

**#include**

**int main()**

**{**

**if(cv::getNumThreads()==4 && cv::checkHardwareSupport(CV\_CPU\_NEON)==1)**

**std::cout << "OpenCV is optimized" << std::endl;**

**return 0;**

**}**

If it doesn't print the message, it means that your OpenCV doesn’t support multi-threading and/or NEON vectorization.

**0- Prepare From Ground Zero**

Ensure that OS is current and updated. I am currently using Raspbian Stretch OS. Open a terminal and run the below:

**sudo apt-get update**

**sudo apt-get upgrade**

Install screen(to ensure process continues to run even if terminal connection is lost) and htop(performance monitoring). These may already be installed on your pi.

**sudo apt-get install screen**

**sudo apt-get install htop**

Prepare your pi for installation, by cleaning up some software

**sudo apt-get -y purge wolfram-engine**

**sudo apt-get -y purge libreoffice\***

**sudo apt-get -y clean**

**sudo apt-get -y autoremove**

**1- Uninstall your current OpenCV**

To build a customized OpenCV with the mentioned capabilities, you must first delete your current version. If you had installed it via sudo apt install libopencv-dev, just run sudo apt purge libopencv-dev. But if you had installed it from source, you must first go to its build folder and then run:

**sudo make uninstall**

**sudo rm -r \***

To make sure that your Pi is clean from any OpenCV lib, run the following commands. If it prints, then there is something wrong with your uninstallation.

**cd /usr/local/lib**

**ls | grep -e libopencv**

**2 - Install an optimized OpenCV**

Before building OpenCV itself, we must install some base dependencies:

**sudo apt update**

**sudo apt upgrade**

**sudo apt install build-essential cmake pkg-config**

**sudo apt install libjpeg-dev libtiff5-dev libpng-dev**

**sudo apt install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev**

**sudo apt install libxvidcore-dev libx264-dev**

**sudo apt install libgtk2.0-dev libgtk-3-dev**

**sudo apt install libatlas-base-dev gfortran**

**Then install** [**Intel TBB**](https://www.threadingbuildingblocks.org/) **and** [**OpenBLAS**](https://github.com/xianyi/OpenBLAS) **via:**

**sudo apt-get install libtbb-dev**

**sudo apt-get install libopenblas-dev liblapacke-dev**

**Install python, pip, numpy and scipy libraries**

**sudo apt-get -y install python3-dev python3-pip**

**sudo pip3 install numpy scipy**

**Now clone OpenCV and OpenCV-Contrib (optional):**

**cd ~**

**mkdir OpenCV && cd OpenCV**

**wget -O opencv.zip https://github.com/opencv/opencv/archive/4.5.0.zip**

**wget -O opencv\_contrib.zip https://github.com/opencv/opencv\_contrib/archive/4.5.0.zip**

**unzip opencv.zip**

**unzip opencv\_contrib.zip**

and finally, run:

**cd opencv-4.5.0**

**mkdir build && cd build**

**cmake -DCMAKE\_BUILD\_TYPE=RELEASE -DCMAKE\_INSTALL\_PREFIX=/usr/local -DWITH\_TBB=ON -DWITH\_LAPACK=ON -DENABLE\_VFPV3=ON -DENABLE\_NEON=ON -DBUILD\_TESTS=OFF -DINSTALL\_C\_EXAMPLES=OFF -DINSTALL\_PYTHON\_EXAMPLES=OFF -DBUILD\_EXAMPLES=OFF -DOPENCV\_EXTRA\_MODULES\_PATH=~/OpenCV/opencv\_contrib-4.5.0/modules ..**

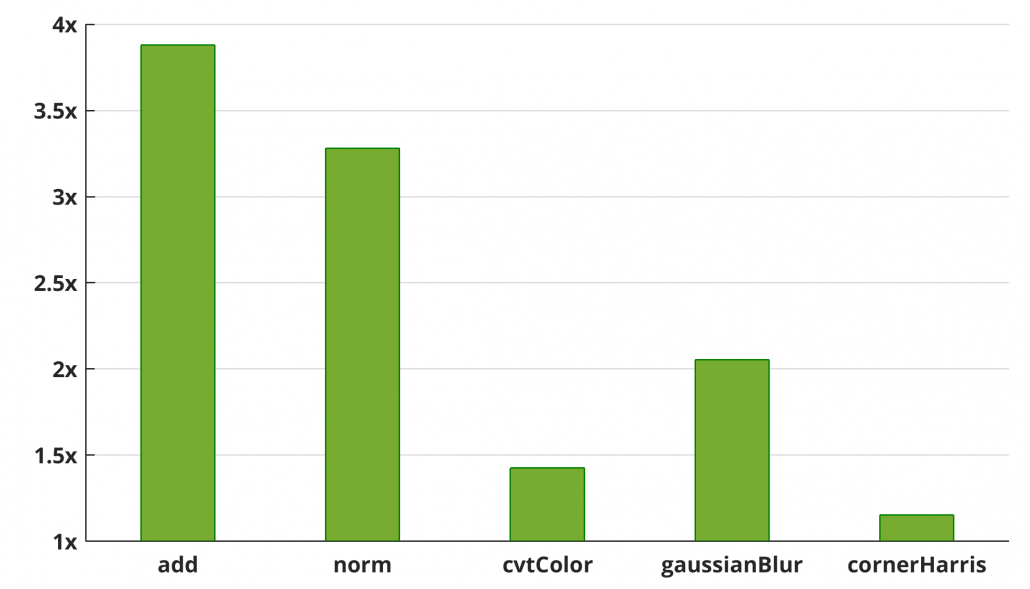
**make -j2**

**sudo make install**

**sudo ldconfig**

The flag **–DWITH\_TBB** enables multi-threading, while the flag **-DWITH\_LAPACK** enables faster matrix operations with OpenBLAS. The **–DENABLE\_VFPV3=ON and –DENABLE\_NEON=ON** flags enable OpenCV to use the NEON coprocessor for vectorization. Other flags just turn off building tests and example programs to reduce the overall OpenCV building time.

After building, you have an OpenCV that certainly works faster. The following figure shows the boost-up of some of the well-known functions.



**3- Notes**

If you are using a 64bit OS on your Pi, then don't **set  –DENABLE\_VFPV3=ON** and **–DENABLE\_NEON=ON**. It throws an error. CMake itself will find the co-processors automatically (tested with OpenCV 4.2.0 on Ubuntu Mate 18).

Although the [documents](https://docs.opencv.org/3.4.1/d7/dff/tutorial_how_to_use_OpenCV_parallel_for_.html) suggest more threading frameworks like OpenMP, pthreads, Concurrency, and GCD, I didn’t experience a multi-threaded OpenCV after setting –DWITH\_OPENMP=ON.

You can also use [libjpeg-turbo](https://libjpeg-turbo.org/) instead of OpenCV’s default libjpeg library. It uses NEON instructions to boost up Jpeg reading and writing. If you are interested, you must first build it by:

**wget https://github.com/libjpeg-turbo/libjpeg-turbo/archive/1.5.0.tar.gz -O libjpeg-turbo.tar.gz**

**tar xvf libjpeg-turbo.tar.gz**

**cd libjpeg-turbo**

**mkdir build**

**autoreconf –fiv**

**cd build**

**sh /configure**

**make -j4**

**sudo make install**

and then add the following statements to your CMake command:

**-DWITH\_JPEG=ON -DBUILD\_JPEG=OFF -DJPEG\_INCLUDE\_DIR=/opt/libjpeg-turbo/include/ -DJPEG\_LIBRARY=/opt/libjpeg-turbo/lib32/libjpeg.a**

The instructions mentioned in this post are not limited to Raspberry Pi. They are applicable to all SBCs equipped with multi-core or newer processors that support Neon technology, including Odroid XU4, BeagleBoard X15, Nvidia Jetson series, etc.

**Reference Links**

1. [**https://www.nerdynat.com/programming/2019/how-to-install-opencv-on-raspberry-pi-3b/**](https://www.nerdynat.com/programming/2019/how-to-install-opencv-on-raspberry-pi-3b/)
2. [**http://imrid.net/?p=3917**](http://imrid.net/?p=3917)