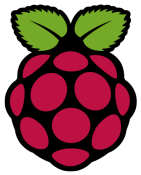
## Raspberry Pi  Installation OpenCV3 sur Jessy



### *1 procédure  préliminaire*

Votre carte mémoire doit avoir au minimum une capacité de **8Go  (16Go c'est mieux)**. La première chose à faire est de mettre à niveau le système.

```
sudo apt-get update
sudo apt-get upgrade
sudo rpi-update
```

Vous avez besoin de redémarrer la rpi après la mise à jour du firmware. En Juillet 2017 le noyau de linux se remet également à jour.

```
sudo reboot
```

Maintenant nous devons installer les outils de développement.

```
sudo apt-get install build-essential git cmake pkg-config
```

Puis, nous pouvons passer à l'installation de paquets d'E/S d'image qui nous permettent de charger les formats de fichiers image tels que JPEG, PNG, TIFF, etc...

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

Nous avons également besoin de paquets d'E/S vidéo. Ces paquets nous permettent de charger différents formats de fichiers vidéo ainsi que de travailler avec des flux vidéo:

```
sudo apt-get install libavcodec-dev libavformat-dev sudo
sudo apt-get install libswscale-dev libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
```



Nous devons installer la bibliothèque de développement GTK afin que nous puissions compiler le sous-module Highgui d'OpenCV, qui nous  permet d'afficher des images sur notre écran et de créer des interfaces GUI simples:

```
sudo apt-get install libgtk2.0-dev
```

Diverses opérations à l'intérieur d'openCV (telles que les opérations matricielles) peuvent être optimisées en utilisant les dépendances ajoutées suivantes:

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

Enfin, nous devrons installer les fichiers d'en-tête Python 2.7 et Python 3 afin que nous puissions compiler les liens OpenCV avec Python:

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

### *2 Obtenir le code source OpenCV*

```
wget -O opencv.zip
https://github.com/Itseez/opencv/archive/3.1.0.zip
```

décompresser l'archive **(sachez que la version 3.2.0 est également disponible)**

```
unzip opencv.zip
```

Pour l'installation complète d'OpenCV 3 (qui comprend des fonctionnalités telles que SIFT et SURF), assurez-vous de récupérer également le dépo opencv_contrib.

```
wget -O opencv_contrib.zip
https://github.com/Itseez/opencv_contrib/archive/3.1.0.
zip

unzip opencv_contrib.zip
```

L'installation de ces paquets n'est certainement pas nécessaire pour ouvrir et exécuter OpenCV avec Python sur votre Raspberry Pi, mais je vous recommande vivement de les installer!

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

sudo python get-pip.py
sudo pip install virtualenv virtualenvwrapper
sudo rm -rf ~/.cache/pip
```

Une fois que virtualenv et virtualenvwrapper ont été installés, nous devons mettre à jour notre fichier ~ / .profile et insérer les lignes suivantes en bas du fichier.

```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

Cette commande va créer un nouvel environnement virtuel Python nommé cv en utilisant Python 3.

```
mkvirtualenv cv -p python3
```

Ouvrez un nouveau terminal, vous devrez utiliser la commande workon pour accéder à l'environnement virtuel cv.

```
$ source ~/.profile

$ workon cv
```

Comment vérifier si vous êtes dans l'environnement virtuel "cv"

Pour valider et vous assurer que vous êtes dans l'environnement virtuel cv, examinez votre ligne de commande - si vous voyez le texte (cv) précédant votre invite, vous êtes dans l'environnement virtuel cv:

Installation de NumPy sur votre Raspberry Pi



En supposant que vous êtes dans l'environnement virtuel cv, nous pouvons installer NumPy, une dépendance importante lors de la compilation des liaisons Python pour OpenCV.

```
$ pip install numpy
```

Vous voudrez peut-être prendre une tasse de café ou faire une promenade pendant que NumPy se télécharge et s'installe:

## *3 Compilation et installation d'OpenCV*

Avant de passer à l'étape de la compilation, assurez-vous d'examiner la sortie de Cmake!

Faites défiler la partie intitulée Python 2 et Python 3.

Si vous compilez OpenCV 3 pour Python 3, vous devez vous assurer que la section Python 2 ressemble à celle-ci surlignée en jaune:

```
--   Python 2:
--     Interpreter:                 /usr/bin/python2.7 (ver 2.7.9)
--     Libraries:                   /usr/lib/arm-linux-gnueabihf/libpython2.7.so (ver 2.7.9)
--     numpy:                       /usr/lib/python2.7/dist-packages/numpy/core/include (ver
1.8.2)
--     packages path:               lib/python2.7/dist-packages
--
--   Python 3:
--     Interpreter:                 /home/pi/.virtualenvs/cv/bin/python3.4 (ver 3.4.2)
--     Libraries:                   /usr/lib/arm-linux-gnueabihf/libpython3.4m.so (ver 3.4.2)
--     numpy:                       /home/pi/.virtualenvs/cv/lib/python3.4/site-
packages/numpy/core/include (ver 1.13.0)
--     packages path:               lib/python3.4/site-packages
```

Enfin, nous sommes maintenant prêts à compiler OpenCV: pour cela on utilisera les quatre cœurs du processeur avec l'option -j4 . Mon expérience personnelle m'a montré qu'il faut compiler en étant connecté en SSH. Donc déconnecté du bureau graphique.

```
make -j4
```

Une fois que la compilation d'OpenCV 3 est terminé, votre affichage devrait ressembler à l'écran suivant:

```
[100%] Linking CXX executable ../../bin/cpp-tutorial-Remap_Demo
[100%] Built target tutorial_Remap_Demo
[100%] Linking CXX executable ../../bin/cpp-tutorial-Sobel_Demo
[100%] Linking CXX executable ../../bin/cpp-tutorial-calcBackProject_Demo1
[100%] Built target tutorial_Sobel_Demo
[100%] Linking CXX executable ../../bin/cpp-tutorial-copyMakeBorder_demo
[100%] Built target tutorial_calcBackProject_Demo1
[100%] Built target tutorial_copyMakeBorder_demo
(cv) pi@raspberrypi3bis:~/opencv-3.0.0/build $
```

Cependant, en raison des conditions de vitesse, il existe des moments où des erreurs peuvent se produire lors de la compilation avec plusieurs noyaux. Si cela vous arrive, je vous suggère de recommencer la compilation et de n'utiliser qu'un seul noyau. (temps de compilation environ deux heures trente)

A partir de maintenant, vous devez installer OpenCV 3 sur votre Raspberry Pi3:

```
sudo make  install
sudo ldconfig
```

Nous avons presque terminé - juste quelques étapes à suivre et vous serez prêt à utiliser votre Raspberry Pi 3 avec OpenCV 3.1.0

```
(cv) pi@raspberrypi3bis:~/opencv-3.0.0/build $ ls -l
/usr/local/lib/python3.4/site-packages/
total 1458
-rw-r--r-- 1 root staff 1461584 juil.  4 17:42 cv2.cpython-34m.so
```

Je ne sais honnêtement pas pourquoi, c'est peut-être un bug dans le script CMake, mais lors de la compilation des liaisons OpenCV 3 pour Python 3+, le fichier .so de sortie est nommé cv2.cpython-34m.so  plutôt que simplement cv2.so (comme dans les liaisons avec Python 2.7).



Pour corriger ce bug, la  solution la plus simple  est de renommer le fichier:

```
cd /usr/local/lib/python3.4/site-packages/
sudo  mv cv2.cpython-34m.so cv2.so
```

Après avoir renommé vers cv2.so, nous pouvons associer nos liens OpenCV à l'environnement virtuel cv pour Python 3.4:

```
cd ~/.virtualenvs/cv/lib/python3.4/site-packages/
ln -s /usr/local/lib/python3.4/site-packages/cv2.so cv2.so
```

## *4 Vérifier l'installation avec python*

```
(cv)pi@raspberrypi3bis:~/.virtualenvs/cv/lib/python3.
4/site-packages $ python
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license" for
more information.
>>> import cv2
>>> cv2.__version__
'3.0.0'
>>>
```

## *4 Vérifier l'installation avec C++*

Vous devriez pouvoir compiler et exécuter le programme très simple suivant

```
#include <opencv2/opencv.hpp>

using namespace cv;

int main(int argc, char **argv)
{
   // Rectangle vert 300 par 200
   Mat img(200,300,CV_8UC3, Scalar(0,255,0));
   imshow("Test", img);
   // Attente appui sur une touche
   waitKey(0);
   return 0;
}
```

Pour compiler ce programme, vous pouvez utiliser la ligne de commande suivante :

```
g++ $(pkg-config --libs --cflags opencv) -o test  test.cpp
```

si l'installation est correct vous obtenez un exécutable test
résultat attendu : affichage d'un rectangle vert dans une fenêtre à l'écran :
Vous pouvez également vérifier le résultat de la commande echo suivante :

```
pi@raspberrypi3:~/openCV/testInstallation $ echo $(pkg-config --libs --cflags opencv)
-I/usr/local/include/opencv -I/usr/local/include -L/usr/local/lib
-lopencv_stitching -lopencv_superres -lopencv_videostab -lopencv_aruco
-lopencv_bgsegm -lopencv_bioinspired -lopencv_ccalib -lopencv_dnn
-lopencv_dpm -lopencv_fuzzy -lopencv_line_descriptor -lopencv_optflow
-lopencv_plot -lopencv_reg -lopencv_saliency -lopencv_stereo
-lopencv_structured_light -lopencv_rgbd -lopencv_surface_matching
-lopencv_tracking -lopencv_datasets -lopencv_text -lopencv_face
-lopencv_xfeatures2d -lopencv_shape -lopencv_video -lopencv_ximgproc
-lopencv_calib3d -lopencv_features2d -lopencv_flann -lopencv_xobjdetect
-lopencv_objdetect -lopencv_ml -lopencv_xphoto -lopencv_highgui
-lopencv_videoio -lopencv_imgcodecs -lopencv_photo -lopencv_imgproc
-lopencv_core
```

L'installation est terminée. Vous pouvez maintenant passer à l'installation du driver V4L. (voir fiche suivante).



Fiche 40 Raspberry  Installation d'openCV 3.1.0