OPENCV 库 ubuntu 下安装

一、opencv for c/c++库安装

参考 http://rodrigoberriel.com/2014/10/installing-opencv-3-0-0-on-ubuntu-14-04/

- 进入 linux 命令行
- 安装 opencv 所依赖的库(可直接输入下面命令)

sudo apt-get -y install libopencv-dev build-essential cmake git libgtk2.0-dev pkg-config python-dev python-numpy libdc1394-22 libdc1394-22-dev libjpeg-dev libpng12-dev libtiff4-dev libjasper-dev libavcodec-dev libavformat-dev libswscale-dev libxine-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev libtbb-dev libqt4-dev libfaac-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev x264 v4l-utils unzip

- 下载 opencv 源码(git 网站下载)

  在自己的 git 仓库下创建 opencv 目录，并下载 opencv 源码

  mkdir opencv

  cd opencv

  wget https://github.com/Itseez/opencv/archive/3.0.0-alpha.zip -O opencv-3.0.0-alpha.zip

  unzip opencv-3.0.0-alpha.zip

- 安装 opencv

  cd opencv-3.0.0-alpha

  mkdir build

  cd build

  cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D WITH_TBB=ON -D WITH_V4L=ON -D WITH_QT=ON -D WITH_OPENGL=ON ..

  make -j $(nproc)

  sudo make install

  Ps: 编译一包代码用的命令是 make, 而通常有一些代码包很大，几百 M 甚至几个 G，例如安卓 5.0 的源代码有 10G 多，这个时候如果只执行 make，编译可能会耗时几天甚至更长时间，但是 linux 下提供了多线程编译，具体要看源码包的 Makefile 是否支持多线程，多线程指的是同时开多个通道一起编译源代码，所使用的命令是 make -j X; 多线程是以牺牲电脑或者服务器的资源为代价来提示速度， 线程越多，make 的 CPU 使用率越高，做其他事情也就会越卡，通常视电脑性能而定 一般可以 make -j8 或者 make -j16， 好一点的服务器可使用 make -j64 或 make -j128。 (不一定是 2 的整数倍，也可以 make -j100) make -j$(nproc) 具体什么意思，暂时不清楚，以后看看。。

- 编译完成，配置 opencv 可以正常工作

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

- 至此，已经完成 opencv 的安装工作，尝试一把 opencv 自带例程
  例如人脸识别例程
  - 开始编译 samples...
    cd opencv/opencv-3.0.0-alpha/samples/
    sudo cmake .
    sudo make -j $(nproc)

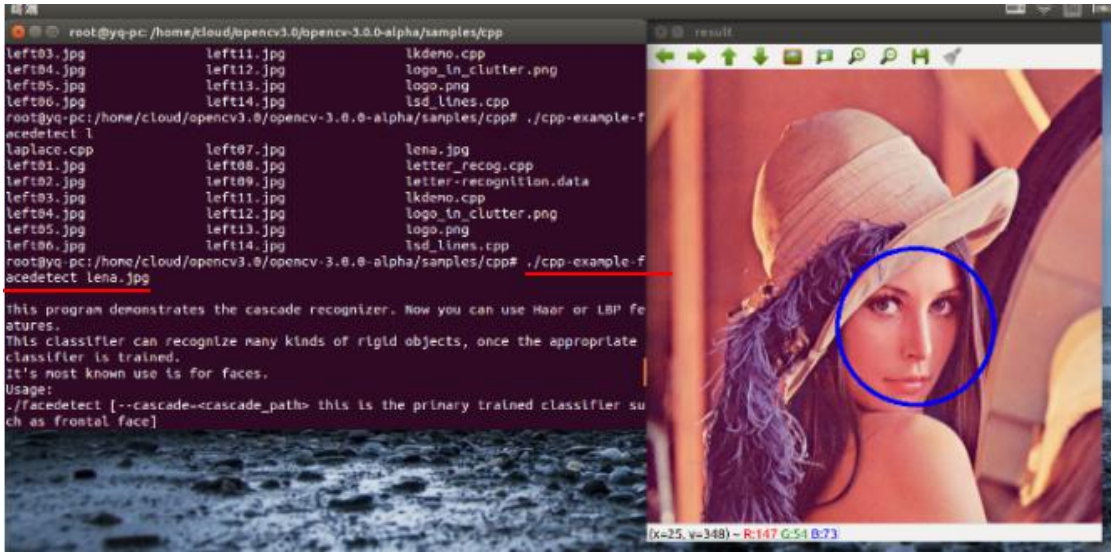
```
root@yq-pc:/home/cloud/opencv3.0/opencv-3.0.0-alpha#
root@yq-pc:/home/cloud/opencv3.0/opencv-3.0.0-alpha# cd samples/
android/              cpp/                  python2/
CMakeCache.txt        directx/              tapi/
CMakeFiles/           gpu/                  winrt/
cmake_install.cmake   java/
CMakeLists.txt        Makefile
root@yq-pc:/home/cloud/opencv3.0/opencv-3.0.0-alpha# cd samples/
root@yq-pc:/home/cloud/opencv3.0/opencv-3.0.0-alpha/samples# sudo cmake .
-- Configuring done
-- Generating done
-- Build files have been written to: /home/cloud/opencv3.0/opencv-3.0.0-alpha/sa
mples
root@yq-pc:/home/cloud/opencv3.0/opencv-3.0.0-alpha/samples# make -j$(nproc)
Linking CXX executable cpp-example-3calibration
Linking CXX executable cpp-example-bgfg_segm
Linking CXX executable cpp-example-calibration
Linking CXX executable cpp-example-camshiftdemo
[  0%] Built target example_3calibration
Linking CXX executable cpp-example-cloning_demo
[  2%] [  2%] [  3%] Built target example_bgfg_segm
Built target example_calibration
Built target example_camshiftdemo
```

  - 编译完成

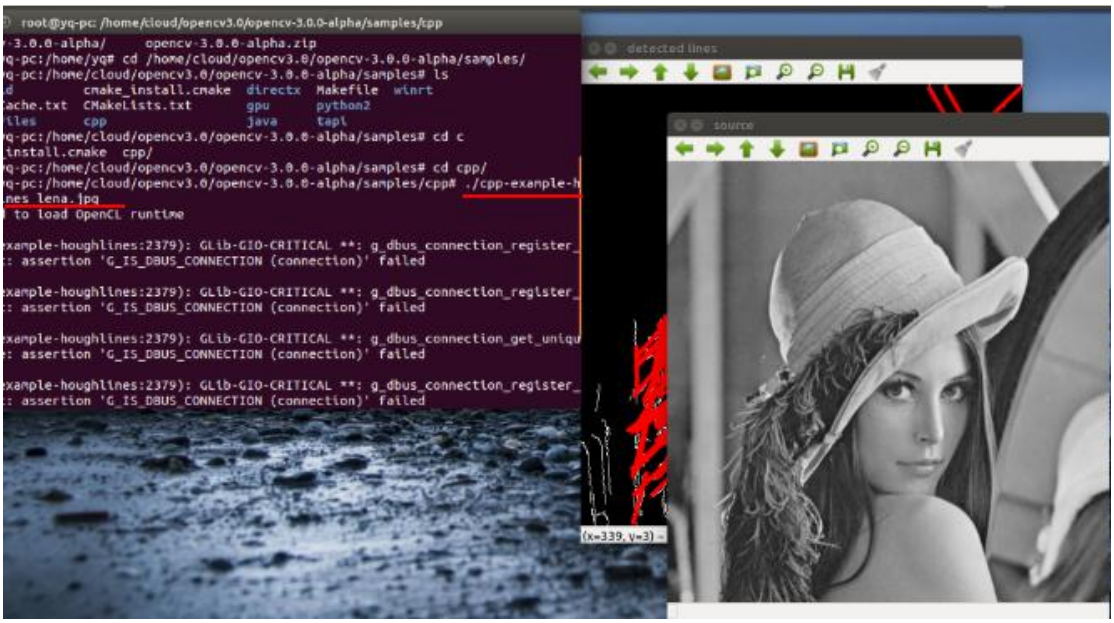
```
root@yq-pc: /home/cloud/opencv3.0/opencv-3.0.0-alpha/samples
[ 90%] Built target tutorial_hull_demo
Linking CXX executable cpp-tutorial-mat_mask_operations
Linking CXX executable cpp-tutorial-mat_the_basic_image_container
[ 91%] Built target tutorial_introduction_to_svm
Linking CXX executable cpp-tutorial-moments_demo
[ 92%] Built target tutorial_introduction_windows_vs
[ 93%] Linking CXX executable cpp-tutorial-non_linear_svms
Built target tutorial_mat_the_basic_image_container
Linking CXX executable cpp-tutorial-npr_demo
[ 93%] Built target tutorial_mat_mask_operations
Linking CXX executable cpp-tutorial-objectDetection
[ 94%] Built target tutorial_moments_demo
[ 95%] Built target tutorial_npr_demo
Linking CXX executable cpp-tutorial-objectDetection2
[ 96%] Built target tutorial_non_linear_svms
Linking CXX executable cpp-tutorial-video-input-psnr-ssim
Linking CXX executable cpp-tutorial-pointPolygonTest_demo
[ 96%] Built target tutorial_objectDetection
Linking CXX executable cpp-tutorial-video-write
[ 97%] Built target tutorial_video-input-psnr-ssim
[ 98%] Built target tutorial_objectDetection2
[ 99%] Built target tutorial_pointPolygonTest_demo
[100%] Built target tutorial_video-write
root@yq-pc:/home/cloud/opencv3.0/opencv-3.0.0-alpha/samples#
```

  - 运行人脸识别例程
    cd cpp/
    ./cpp-example-facedetect lena.jpg
    Ps: // (../data/lena.jpg) OpenCV 3.0 beta 图片有可能在./data/下

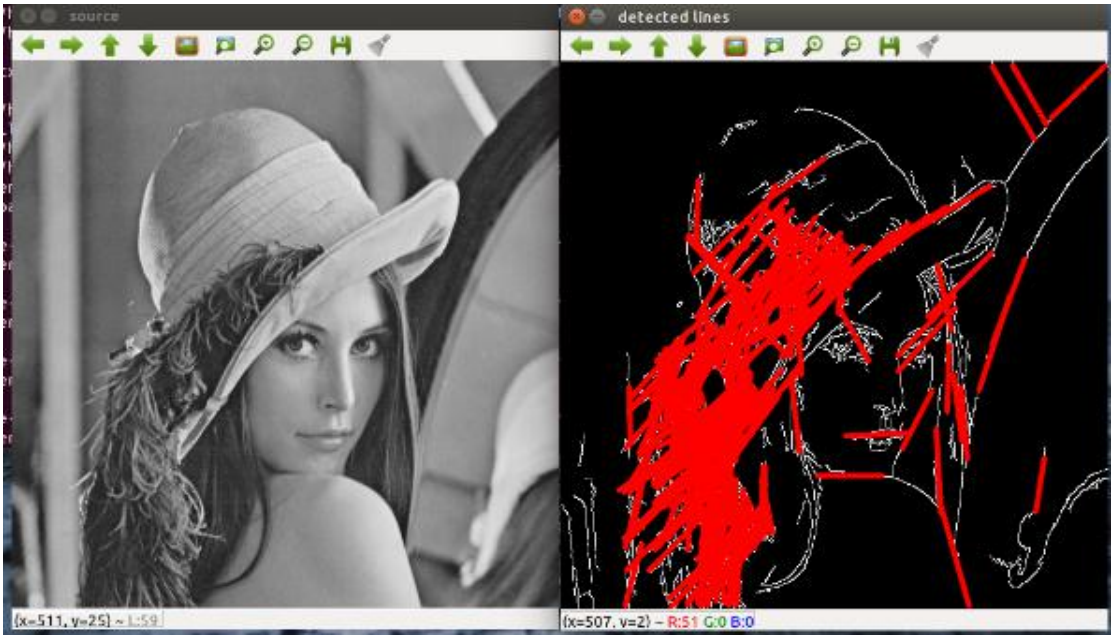
运行成功则会在屏幕上显示 lena 的图片，按 enter 键即可退出。



- 运行线检测例程
  ./cpp-example-houghlines lena.png
  Ps: // (../data/lena.jpg) OpenCV 3.0 beta 图片有可能在./data/下



图片效果对比图如下：



- 运行自己的 opencv 程序

Step 1. 配置 opencv lib 调用路径

调用路径的配置前提条件是已经执行了编译 opencv 的工作，有两种配置方式可以使例程调用 opencv 的 lib

1、直接将 lib 移动至系统默认的 lib 路径，同时配置 include 路径

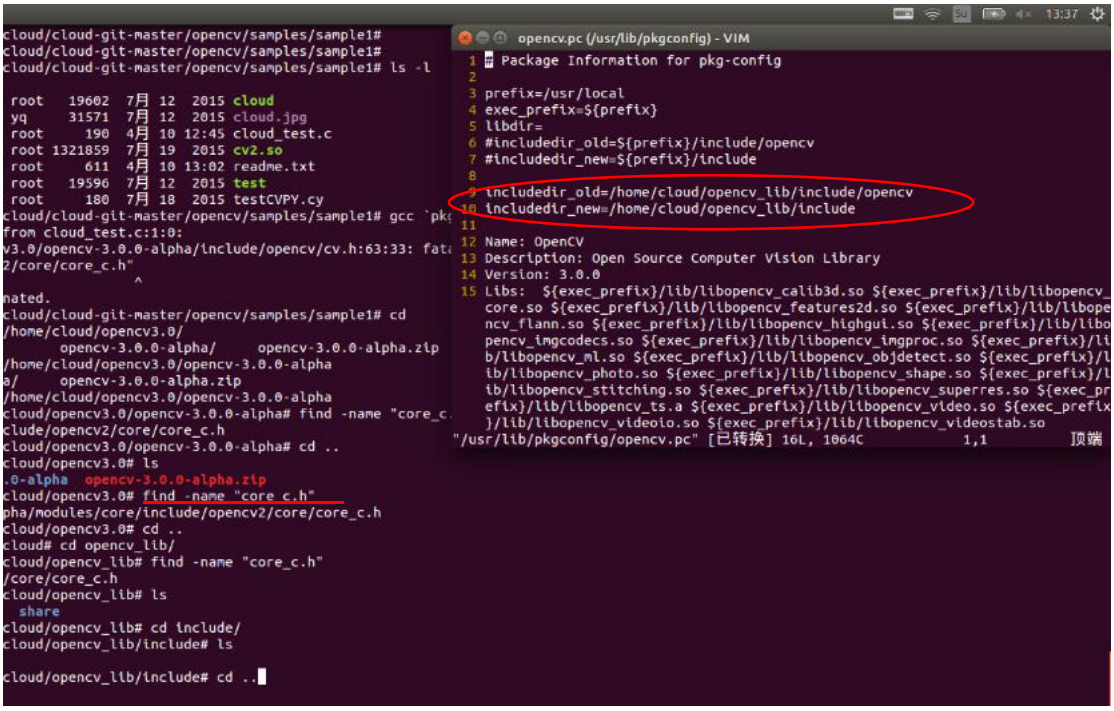
```
root@yq-pc:/home/cloud/opencv_lib# find -name "opencv.pc"
./lib/pkgconfig/opencv.pc
root@yq-pc:/home/cloud/opencv_lib# cd ..
root@yq-pc:/home/cloud# cd opencv
opencv3.0/  opencv_lib/
root@yq-pc:/home/cloud# cd opencv3.0/
root@yq-pc:/home/cloud/opencv3.0# find -name "opencv.pc"
./build/unix-install/opencv.pc
./opencv-3.0.0-alpha/release/unix-install/opencv.pc
root@yq-pc:/home/cloud/opencv3.0# cp opencv-3.0.0-alpha/release/unix-install/ope
ncv.pc /usr/lib/pkgconfig/
```

✓ 在创建的 opencv 路径下使用 find -name “opencv.pc” 命令找到 opencv.pc 文件的路径，路径是自己定的，每个人路径不一样

✓ 将该文件 cp 到系统默认路径

cp opencv-3.0.0-alpha/release/unix-install/opencv.pc /usr/lib/pkgconfig

✓ 修改 opencv.pc 内容设置 include 文件路径

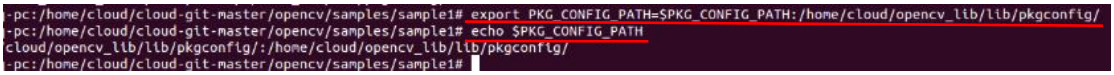


Include 文件路径同样可以用 find 命令来确定：

比如 opencv 中一个头文件 core_c.h

find -name “core_c.h” 找到相关路径后，修改 opencv.pc 里的 includedir_old 和 includedir_new，并退出保存

2、设置环境变量方式也可以配置 include 和 lib



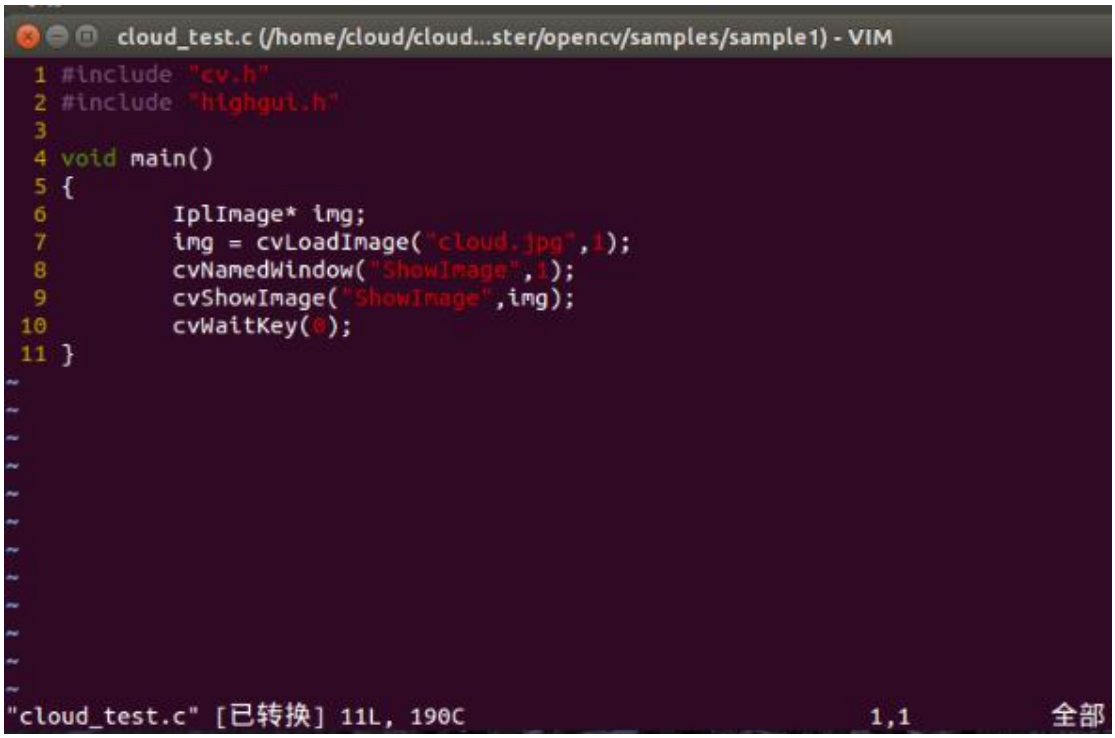
- ✓ 还是通过 find -name “opencv.pc” 命令找到 opencv.pc 文件的路径
- ✓ 然后 export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/xxx (xxx 为 opencv.pc 的路径)

使用 echo 命令可以查看是否有配置到

Step 2. 编写 opencv demo 例程

编写了一个播放一张图片的例程 cloud_test.c，内容如下：

加载一副图片->新建一个显示窗口->显示图片->等待按键响应



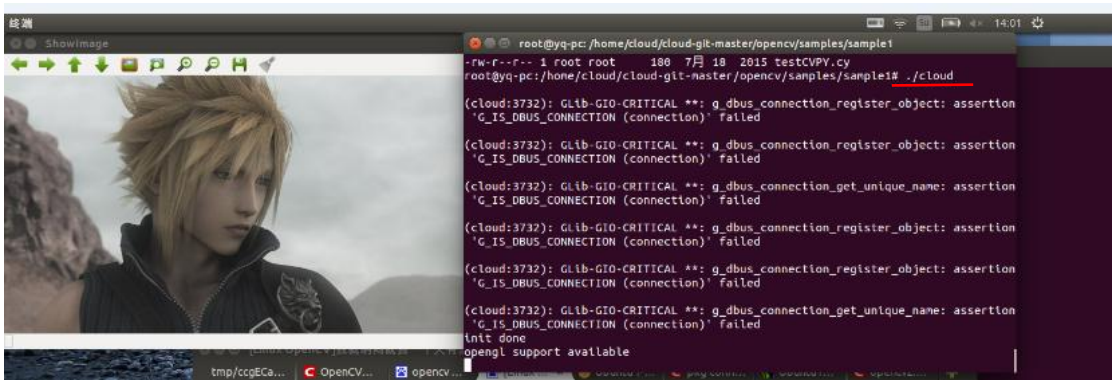
Step 3. 编译该例程文件，生成可执行档

```
root@yq-pc:/home/cloud/cloud-git-master/opencv/samples/sample1# ls -l
总用量 1368
-rwxr-xr-x 1 root root    8682  4月 10 13:59 cloud
-rw-rw-r-- 1 yq   yq     31571  7月 12  2015 cloud.jpg
-rw-r--r-- 1 root root     205  4月 10 13:55 cloud_test.c
-rwxr-xr-x 1 root root 1321859  7月 19  2015 cv2.so
-rw-r--r-- 1 root root     674  4月 10 13:44 readme.txt
-rwxr-xr-x 1 root root   19596  7月 12  2015 test
-rw-r--r-- 1 root root     180  7月 18  2015 testCVPY.cy
root@yq-pc:/home/cloud/cloud-git-master/opencv/samples/sample1# rm -rf cloud
root@yq-pc:/home/cloud/cloud-git-master/opencv/samples/sample1# g++ cloud_test.c -o cloud `pkg-config --libs --cflags opencv`
```

执行 g++ cloud_test.c -o cloud `pkg-config --libs --cflags opencv` 之后
将会把 cloud_test.c 生成一个名为 cloud 的可执行档。
(Ps：写的顺序千万不要弄反，不然会出错，出错如下图)

```
root@yq-pc:/home/cloud/cloud-git-master/opencv/samples/sample1# gcc `pkg-config --cflags --libs opencv` -o cloud cloud_test.c
In file included from cloud_test.c:1:0:
/home/cloud/opencv3.0/opencv-3.0.0-alpha/include/opencv/cv.h:63:33: fatal error: opencv2/core/core_c.h: 没有那个文件或目录
 #include "opencv2/core/core_c.h"
                                 ^
compilation terminated.
root@yq-pc:/home/cloud/cloud-git-master/opencv/samples/sample1#
```

Step 4. 运行该例程可执行档./cloud，可查看效果



二、Opencv for python 库安装

可参考 http://pinkyjie.com/2010/10/19/ubuntu-opencv-python/

- Opencv for python 其实和 opencv for c/c++一样的安装步骤，执行一次就好。只是在 python 环境下会依赖 cv2.so，即执行编译 opencv 后会产生这个 so，可通过 find 命令来查找。
- 新建一个 python opencv 演示例程 testCVPY.cy

```
testCVPY.cy (/home/cloud/cloud-git-master/opencv/samples/sample1) - VIM
 1 import cv2
 2
 3 if __name__ == '__main__':
 4   img = cv2.imread ("cloud.jpg")
 5   cv2.namedWindow ("ShowImage")
 6   cv2.imshow ("ShowImage", img)
 7   cv2.waitKey (0)
 8   cv2.destroyAllWindows()
```

- 将 cv2.so 和 testCVPY.cy 拷贝到同一个目录下，运行 testCVPY.py

```
  -o cloud `pkg-config --libs --cflags opencv`
root@yq-pc:/home/cloud/cloud-git-master/opencv/samples/sample1# ls -l
总用量 1368
-rwxr-xr-x 1 root root    8682  4月 10 14:01 cloud
-rw-rw-r-- 1 yq   yq     31571  7月 12  2015 cloud.jpg
-rw-r--r-- 1 root root     285  4月 10 13:55 cloud_test.c
-rwxr-xr-x 1 root root 1321859  7月 19  2015 cv2.so
-rw-r--r-- 1 root root     674  4月 10 13:44 readme.txt
-rwxr-xr-x 1 root root   19596  7月 12  2015 test
-rw-r--r-- 1 root root     180  7月 18  2015 testCVPY.cy
root@yq-pc:/home/cloud/cloud-git-master/opencv/samples/sample1#
```

直接执行 python testCVPY.cy 提示 cv2 找不到，原因是我的电脑安装的 python 版本是 3.4 的，而 opencv 下载的版本是 for python2.X 的，所以会有问题。

```
-rw-r--r-- 1 root root     674  4月 10 13:44 readme.txt
-rw-r--r-- 1 root root     180  4月 10 14:11 testCVPY.cy
root@yq-pc:/home/cloud/cloud-git-master/opencv/samples/sample1# python testCVPY.cy
Traceback (most recent call last):
  File "testCVPY.cy", line 1, in <module>
    import cv2
ImportError: dynamic module does not define init function (PyInit_cv2)
root@yq-pc:/home/cloud/cloud-git-master/opencv/samples/sample1#
```

如果你的电脑有安装 python2.x 且没有卸载，一般 ubuntu 会默认安装 python2.x，则可直接执行 python2.7 testCVPY.cy ，正常显示出效果。