**Instalación de OpenCV3 + Dlib en Linux (C++ y Python)**

Objetivo:

1. Instalar OpenCV 3.4 en Ubuntu.
2. Instalar y configurar Dlib para trabajar con C++ con Python y Ubuntu
3. Probar la instalación.

Las instrucciones son algo complicadas así que tomese su tiempo de entender y busque referencias en la web si no entiende alguna situación. Esta es la manera correcta de realizer la instalación de C++, OpenCV y Python. Ayúdese con sus compañeros si tiene algún inconveniente.

**Instalar OpenCV en Ubuntu**

**Paso 0: Completar la tarea de Virtualbox con Ubuntu 16.04 LTS como mínimo**

Puede descargarla de http://releases.ubuntu.com/16.04/

Versión ubuntu 16.04.5 desktop amd64 para 64 bits e i386 para 32 bits

Limpiaremos los directorios de build e install para empezar de una manera límpia.

#Crear una variable para version de OpenCV y el directorio actual

cvVersion="3.4.3"

cwd=$(pwd)

# Limpiar los directories build (puede no existir y dar error)

rm -rf opencv/build

rm -rf opencv\_contrib/build

**Paso 1: Actualzación de Paquetes**

sudo apt -y update

sudo apt -y upgrade

**Paso 2: Instalación de Librerías del Sistema Operativo**

# remover cualquier instalación previa de x264

sudo apt -y remove x264 libx264-dev

# ahora instalaremos las dependencias

sudo apt -y install build-essential checkinstall cmake pkg-config yasm

sudo apt -y install git gfortran

sudo apt -y install libjpeg8-dev libjasper-dev libpng12-dev

sudo apt -y install libtiff5-dev

sudo apt -y install libtiff-dev

sudo apt -y install libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev

sudo apt -y install libxine2-dev libv4l-dev

cd /usr/include/linux

sudo ln -s -f ../libv4l1-videodev.h videodev.h

cd $cwd

sudo apt -y install libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev

sudo apt -y install libgtk2.0-dev libtbb-dev qt5-default

sudo apt -y install libatlas-base-dev

sudo apt -y install libfaac-dev libmp3lame-dev libtheora-dev

sudo apt -y install libvorbis-dev libxvidcore-dev

sudo apt -y install libopencore-amrnb-dev libopencore-amrwb-dev

sudo apt -y install libavresample-dev

sudo apt -y install x264 v4l-utils

# Dependencias opcionales

sudo apt -y install libprotobuf-dev protobuf-compiler

sudo apt -y install libgoogle-glog-dev libgflags-dev

sudo apt -y install libgphoto2-dev libeigen3-dev libhdf5-dev doxygen

**Paso 3: Instalar librerías de Python**

sudo apt -y install python-dev python-pip python3-dev python3-pip

sudo -H pip2 install -U pip numpy

sudo -H pip3 install -U pip numpy

sudo apt -y install python3-testresources

Ahora forzaremos el entorno virtual de desarrollo para la instalación de librerías de python. Generalmente es una Buena práctica en orden de separar su en ambiente global de proyecto y ambiente de desarrollo.

*# instalar ambiente virtual de desarrollo*

sudo -H pip2 install virtualenv virtualenvwrapper

sudo -H pip3 install virtualenv virtualenvwrapper

echo "# Virtual Environment Wrapper" >> ~/.bashrc

echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.bashrc

#source ~/.bashrc

cd $cwd

source /usr/local/bin/virtualenvwrapper.sh

Seguido, crearemos el ambiente virtual de Python, además instalaremos dlib en ambos ambientes.

*############ Para Python 2 ############*

*# Crear el ambiente virtual de desarrollo*

mkvirtualenv cv-py2 -p python2

workon cv-py2

*# ahora instalaremos las librerías en este ambiente virtual*

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

*# quit virtual environment*

deactivate

*######################################*

*############ Para Python 3 ############*

*# Crear ambiente virtual*

mkvirtualenv cv-py3 -p python3

workon cv-py3

*# Ahora instalar las librerías de Python en este ambiente virtual*

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

*# desactivar el ambiente virtual*

deactivate

**Paso 4: Descargar opencv y opencv\_contrib**

Descargaremos opencv and opencv\_contrib de los paquetes repositorios en github.

**Paso 4.1: Descargar OpenCV de Github**

Clonaremos el Github de OpenCV y veremos si es la version 3.4.0, si es una superior no importa, el proceso es el mismo.

echo "Downloading opencv and opencv\_contrib"

git clone https://github.com/opencv/opencv.git

cd opencv

git checkout $cvVersion

cd ..

**Paso 4.2: Descargar opencv\_contrib de Github**

Similarmente descargaremos el repositorio 3.4.0 de opencv\_contrib.

git clone https://github.com/opencv/opencv\_contrib.git

cd opencv\_contrib

git checkout $cvVersion

cd ..

**Paso 5: Compile e Instale OpenCV con los módulos contrib**

**Paso 5.1 : Crear un directorio para compilar el codigo fuente**

echo "================================"

echo "Compiling and installing OpenCV with contrib modules"

cd opencv

mkdir build

cd build

**Paso 5.2 : Correr CMAKE**

cmake -D CMAKE\_BUILD\_TYPE=RELEASE \

-D CMAKE\_INSTALL\_PREFIX=/usr/local \

-D INSTALL\_C\_EXAMPLES=ON \

-D INSTALL\_PYTHON\_EXAMPLES=ON \

-D WITH\_TBB=ON \

-D WITH\_V4L=ON \

-D WITH\_QT=ON \

-D WITH\_OPENGL=ON \

-D OPENCV\_EXTRA\_MODULES\_PATH=../../opencv\_contrib/modules \

-D BUILD\_EXAMPLES=ON ..

**Paso 5.3 : Compile e Instale**

make -j4

sudo make install

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

sudo ldconfig

**Paso 5.4 : Crear un symlink in en el entorno de trabajo virtual**

Dependiendo de la version de Python que posea, los paths pueden ser diferentes. El binario de opencv se llama ‘cv2.so’ y puede estar instalado como en el directorio de los site-packages o el de dist-packages. Use el siguiente commando para encontrar la locación en su máquina si lo desea

find /usr/local/lib/ -type f -name "cv2\*.so"

Debería entonces dar la salida al path similar a alguno de los que listo debajo uno o ambos que son para Python 2.x o Python 3.x.

*############ Para Python 2 ############*

*## dist-packages binario instalado*

/usr/local/lib/python2.6/dist-packages/cv2.so

/usr/local/lib/python2.7/dist-packages/cv2.so

*## site-packages binario instalado*

/usr/local/lib/python2.6/site-packages/cv2.so

/usr/local/lib/python2.7/site-packages/cv2.so

*############ Para Python 3 ############*

*## binario en dist-packages*

/usr/local/lib/python3.5/dist-packages/cv2.cpython-35m-x86\_64-linux-gnu.so

/usr/local/lib/python3.6/dist-packages/cv2.cpython-36m-x86\_64-linux-gnu.so

*## binario en site-packages*

/usr/local/lib/python3.5/site-packages/cv2.cpython-35m-x86\_64-linux-gnu.so

/usr/local/lib/python3.6/site-packages/cv2.cpython-36m-x86\_64-linux-gnu.so

**Verifique nuevamente los commandos del path exacto en su máquina antes de correr los siguientes commandos.**

*# SymLink para Python2 y Python3*

py2binPath=$(find /usr/local/lib/ -type f -name "cv2.so")

py3binPath=$(find /usr/local/lib/ -type f -name "cv2.cpython\*.so")

cd ~/.virtualenvs/cv-py2/lib/python2.7/site-packages/

ln -f -s $py2binPath cv2.so

cd ~/.virtualenvs/cv-py3/lib/python3.5/site-packages/

ln -f -s $py3binPath cv2.so

**Paso 6: Probar OpenCV 3**

Activar el ambiente virtual

*############ Para Python 2 ############*

workon cv-py2

*############ Para Python 3 ############*

workon cv-py3

Abrir IPython y verificar la version de OpenCV

*# open ipython (correr este comando en la terminal)*

ipython

*# importer cv2 e imprimir la version (correr este comando en la terminal)*

import cv2

print(cv2.\_\_version\_\_)

*# Si OpenCV3 es instalado correctamente, debe de tener como salida la version instalada, si opencv es 3.4.0 debe imprimir esto.*

*# Presione CTRL+D para salir de ipython*

Ahora puede salir de la ventana de iPython del ambiente virtual.

deactivate

Siempre utilizar el ambiente virtual de python para correr los scripts de Python.

**Paso 7: Realizando ajustes para Dlib para C++**

Existen dos maneras de utilizar OpenCV en C++, la preferida es con CMake, la otra es desde la lína de commandos con g++. Verificaremos ambas maneras.

**7.1. Usando CMakeLists.txt**

En un sistema amplio de instalación (*instalado en /usr/local*), no se debe de ajustar OpenCV\_DIR. Este es auto detectado por CMAKE. Pero cuando no lo es, ud debe ajustar OpenCV\_DIR como sigue acontinuación.

SET(OpenCV\_DIR <OpenCV\_Home\_Dir>/installation/OpenCV-3.4.3/share/OpenCV/)

Asegurese de reemplazar **OpenCV\_Home\_Dir** con el path correcto. Por ejemplo, en mi caso sería:

SET(OpenCV\_DIR /home/issaiass/OpenCV\_installation/installation/OpenCV-3.4.3/share/OpenCV/)

Una vez ha realizado los cambios en su **CMakeLists.txt**, siga los siguientes pasos debajo.

mkdir build && cd build

cmake ..

cmake --build . --config Release

Esto generará su archivo ejecutable en la carpeta **build**.

**7.1. Usando g++**

Para compilar el archivo en OpenCV-3.4.3 (digamos se llama vision\_por\_pc.cpp), use el siguiente comando. Debemos especificar el path de OpenCV utilizando --libs.

g++ `pkg-config --cflags --libs <OpenCV\_Home\_Dir>/installation/OpenCV-3.4.3/lib/pkgconfig/opencv.pc` vision\_por\_pc.cpp -o vision\_por\_pc

**Paso 8. Probando el código generado en C++**

Utilizando CMake para compilar C++. Al principio es complicado, pero no se preocupe, le ayudaremos en el camino.

1. Descomprimir el archivo suministrado
2. Abrir la terminal e ir a la carpeta donde está el archivo de cpp source/so/cpp. Debería contener un archivo CMakelists.txt file que contiene el detector de marcas faciales.
3. Ejecute los siguientes comandos

# Crear un directorio de compilación

mkdir build

cd build

# Configurar para compilar

cmake ..

# Compilar

cmake --build . --config Release

cd ..

1. Debería producir el ejecutable en la carpeta de build. Si da error que dice que no puede encontrar OpenCV, encones modifique OpenCV\_DIR en el **CMakeLists.txt** como se explicó anteriormente.
2. Correr el ejecutable

./build/facialLandmarkDetector

Ud. debería haber conseguido una salida como la siguiente.

[](http://www.learnopencv.com/wp-content/uploads/2017/12/result-facial-Landmarks.jpg)

**Prueba del código de ejemplo para Python**

1. Del archivo de Proyecto extraigalo si no lo ha realizado.
2. Abra la terminal y cambie al folder de python (source/so/python).
3. Execute el siguiente comando

*# activar el ambiente*

workon cv-py3

*# Ejecutar el código de ejemplo*

python facialLandmarkDetector.py

*# Salir del ambiente virtual*

deactivate

Debería tener una salida como la que se ve a continuación:

[](http://www.learnopencv.com/wp-content/uploads/2017/12/result-facial-Landmarks.jpg)

# Como Utilizar CMake

# Usando CMake para construir proyectos de C++

CMake es una herramienta de compilación cruzada para compilar el código.

## Paso 1: Saber donde se encuentra instalado OpenCV ( ****OpenCV\_DIR**** )

Primero busquemos OpenCV instalado en nuestro sistema. Si el sistema no es capaz de hallar OpenCV degbe añadirlo manualmente del archive CMakelists.txt.

### ****En Ubuntu****

En la terminal:

find /usr/local -name OpenCVConfig.cmake

Debe obtener una salida de la forma:

/usr/local/share/OpenCV/OpenCVConfig.cmake

Lo superior indica donde se ha instalado OpenCV

En **/usr/local/share/OpenCV**

## Paso 2: Escribiendo el archive CMakeLists.txt

Ahora aprenderemos como escribir el archive CMake en su proyecto.

CMake lee la configuración del proyecto (path a los archivos cabeceras y librerias) desde el archive llamado CMakeLists.txt. Cada línea de este archive da una explicación de lo que se escribe en la línea superior.

*############# Contenido de una muestra del archivo CMakeLists.txt #############*

*# Define the minimum CMake version which can execute this file*

cmake\_minimum\_required(VERSION 2.8.12)

*# Da el nombre del proyecto*

PROJECT(myProject)

*## Si la librería está instalada en la máquina le diremos que la encuentre.*

*# CMake busca los archivos OpenCVConfig.cmake y OpenCVConfig-version.cmake para*

*# configurar los ajustes para OpenCV.*

*# Existen dos maneras para ayudar a CMake a encontrar estos archivos de OpenCV.*

*# 1. Definir una variable de entorno llamada OpenCV\_DIR y darle el path de folder*

*# la cual contiene OpenCVConfig.cmake y OpenCVConfig-version.cmake*

*# 2. Ajustar OpenCV\_DIR en CMakeLists.txt file. (Paso 1)*

*# Para la segunda opcion puede tener multiples folders de OpenCV* y apuntar a la

# que necesite

*# Ejemplos de OpenCV\_DIR para Linux*

*# Linux*

*# SET(OpenCV\_DIR /usr/local/share/OpenCV/)*

*# Buscar OpenCV*

find\_package( OpenCV REQUIRED )

*# Incluir los archivos headers para OpenCV*

include\_directories( ${OpenCV\_INCLUDE\_DIRS})

*# Incluir cmake file de Dlib. Tiene la configuraci[on para construer Dlib.*

*# con su codigo. Si observa este archive, tiene una similitude en sintaxis a*

*# CMakeLists.txt.*

include(path\_to\_dlib-19.9/dlib/cmake)

*# Ahora suponga tenemos un archive cpp llamado codigo.cpp*

*# Escribiremos una regla para compilar y linkear*

*# En ADD\_EXECUTABLE primero ‘codigo’ es el nombre del ejecutable*

*# La cual será generada cuando se compile ‘codigo.cpp’*

ADD\_EXECUTABLE(myCode myCode.cpp)

*# Para vincular OpenCV\_LIBS y dlib:dlib flags.*

TARGET\_LINK\_LIBRARIES(myCode ${OpenCV\_LIBS} dlib:dlib)

*# Esto es todo lo que necesitamos para constrir nuestro archive de C++.*

*# Pero que pasa si tenemos muchos archivos de C++ para compilar?*

*.*

*# Podemos escribir un macro para no repetir esta tarea indefinidamente.*

*# Define un macro add\_example a nuestras reglas de compilación.*

MACRO(add\_example name)

ADD\_EXECUTABLE(${name} ${name}.cpp)

TARGET\_LINK\_LIBRARIES(${name} ${OpenCV\_LIBS} dlib:dlib)

ENDMACRO()

*# Ahora podemos utilizar este macro para compilar nuestros archivos.*

add\_example(codigo1)

add\_example(codigo2)

## Paso 3: Construir el proyecto utilizando CMake

Compilemos utilizando CMakeLists.txt con el archive escrito.

*# Para compilar codigo.cpp, codigo1.cpp y codigo2.cpp, debe salvar*

*# CMakeLists.txt para escribir en el mismo folder donde*

*# se ha tenido los archivos cpp.*

*# Ahora al abrir la terminal y correr el sigueinte comando para compilar*

mkdir build

cd build

*## Linux*

*# Decirle a CMake que configure y genere los archivos*

*# .. denotar que CMake debe buscar para CMakeLists.txt en*

*# el folder superior de este directorio*

*# Para Linux cmake detecta y usa el toolchain c++ instalado en el sistema.*

*# Debería ser un toolchain gcc.*

cmake ..

*# Ahora decirle a CMake construer el proyecto en Release mode*

*# Tambien podemos usar la configuracion de Debug*

cmake --build . --config Release

*# Cuando el build esta complete, los ejecutables se generaran en*

*# build\Release o en el folder build asi mismo*

*# Ahora al ir hacia el directorio de proyecto*

*# (directorio superior parent del build)*

*# Este es el importante si tienes paths relativos en el codigo*

*# Puede correr los ejecutables ahora.*

./build/release/myCode1

./build/release/myCode2