Laboratorio 3 – Instalación de OpenCV para C++ Python, PC y para Raspberry Pi

Para las siguiente secciones utilice:

* La sección I
  + I.a si posee una máquina con SO Windows
  + I.b si posee una máquina con SO MacOS
* La sección II para Raspberry Pi.

**Objetivo: 100% - Todos deben traer instalado OpenCV para sus computadoras y para el Raspberry pi**

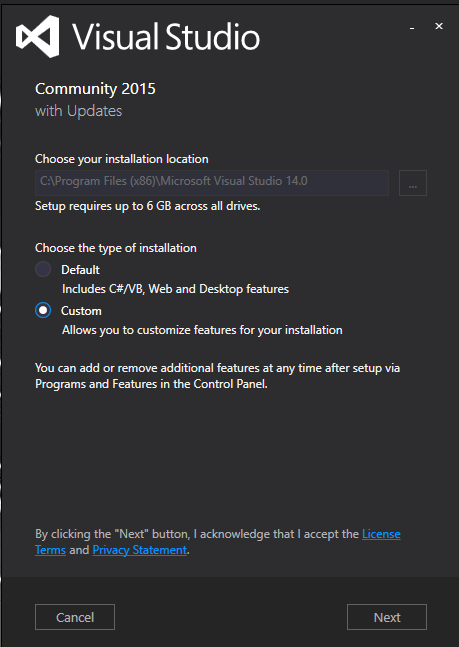
I – Instalación de OpenCV (Open Computer Vision – Librerías de Vision por Computador)

I.a – Instalación de OpenCV en Windows (C++ y Python)

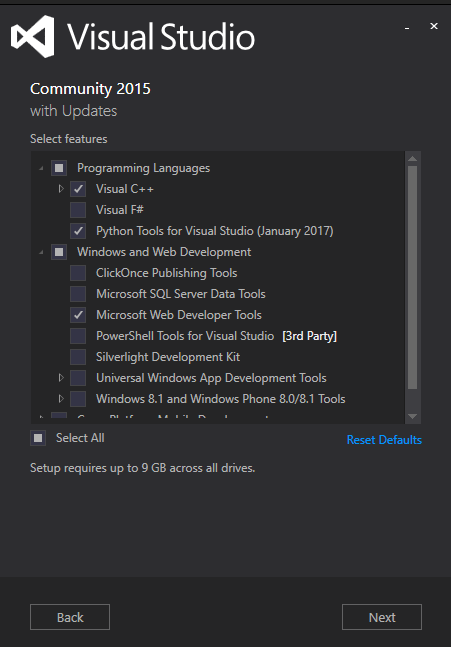
# Utilizaremos CMAKE para construir librerías para utilizar el compilador de Visual Studio. Instalaremos los paquetes de la librería opencv\_contrib. Se puede utilizar como ventana de comandos la genérica de windows o windows powershell indistintamente.

Paso 1 – Instalación de Visual Studio.

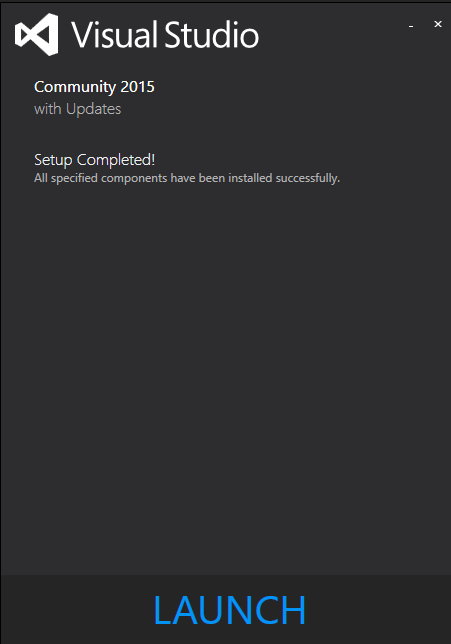
Descargar Visual Studio 2015 community edition <https://www.visualstudio.com/vs/older-downloads/>. Si posee dificuItad para esto use el siguiente vínculo <https://my.visualstudio.com/Downloads?pid=2086>. Si ud. No posee Visual Studio Dev Essentials como cuenta, cree una y regístrese. Correr el instalador, seleccionar “Custom” y en “type of installation” como se observa en la figura inferior.



En la siguiente pantalla seleccione Visual C++ y Python tolos para visual Studio, presionar siguiente.

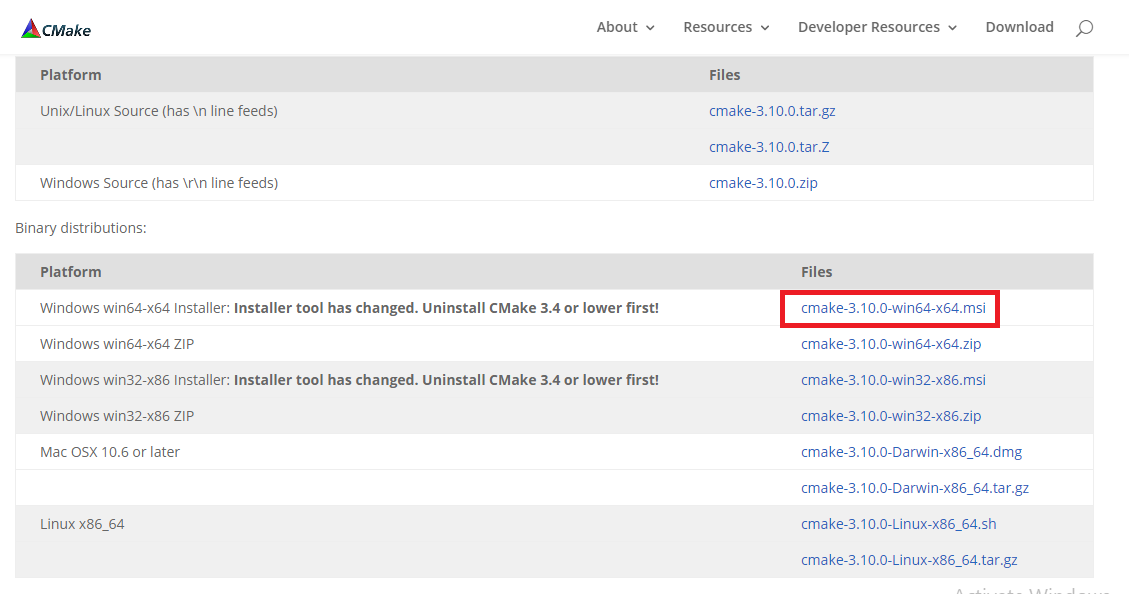


Hacer click en siguiente. Tomará un tiempo en completar la instalación.

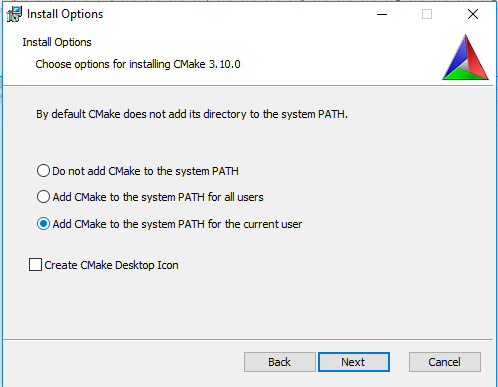


Paso 2 – Instalar CMake (puede descargar la versión mencionada o utilizar la ultima versión)

Descargar e instalar CMake 3.10.0 de <https://cmake.org/download/>



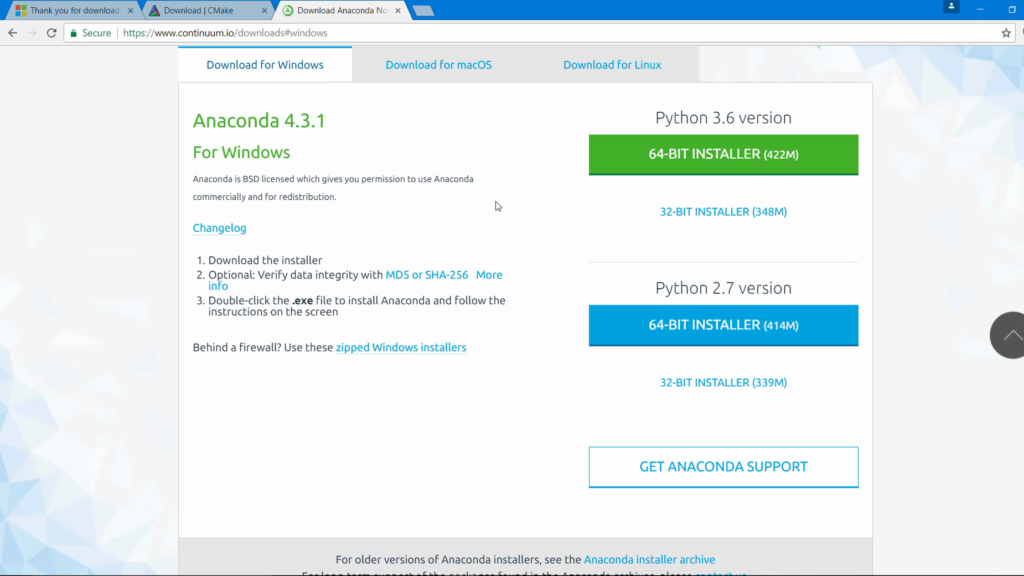
Durante la instalación seleccione “Añadir CMake al system PATH”



Paso 3 – Instalación de Anaconda (una distribución de python)

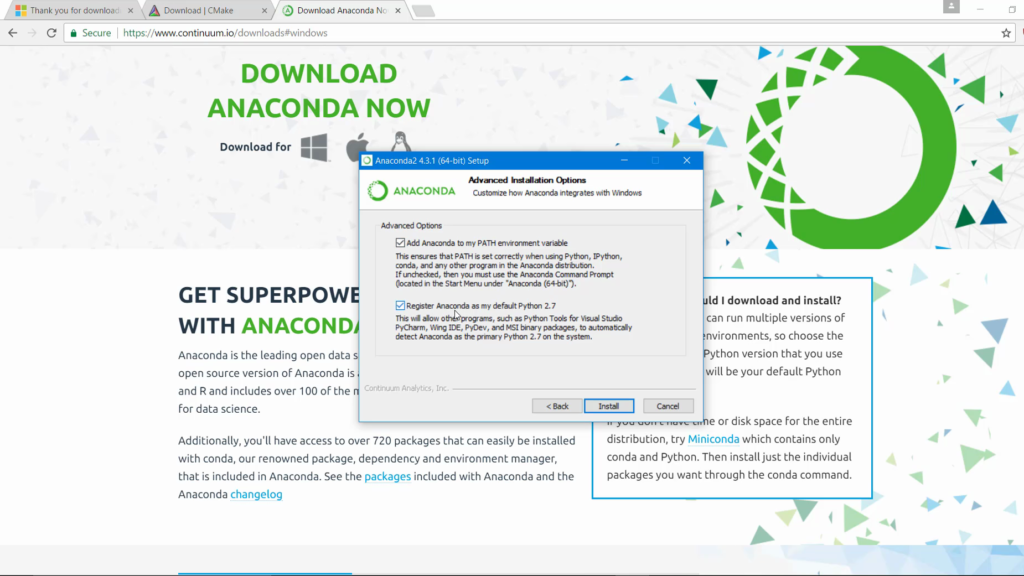
Descargar e instalar Anaconda 64-bits de <https://www.continuum.io/downloads>.

NOTA: Dlib se envía como binaro preconfigurado para Pytho3 y no para Pytho2. Construir Dlib con python bindings desde la fuente puede tardar en realizarse esta operación. Así, es recomendado instalar Anaconda para python3. En caso de que ud. Quiere instalar opencv python bindings para Python2 ud puede instalar Anacoda 2 pero no podrá utilizar Dlib en Python2.



Mientras instala Anaconda, asegurese de que ud. Posee las siguientes partes

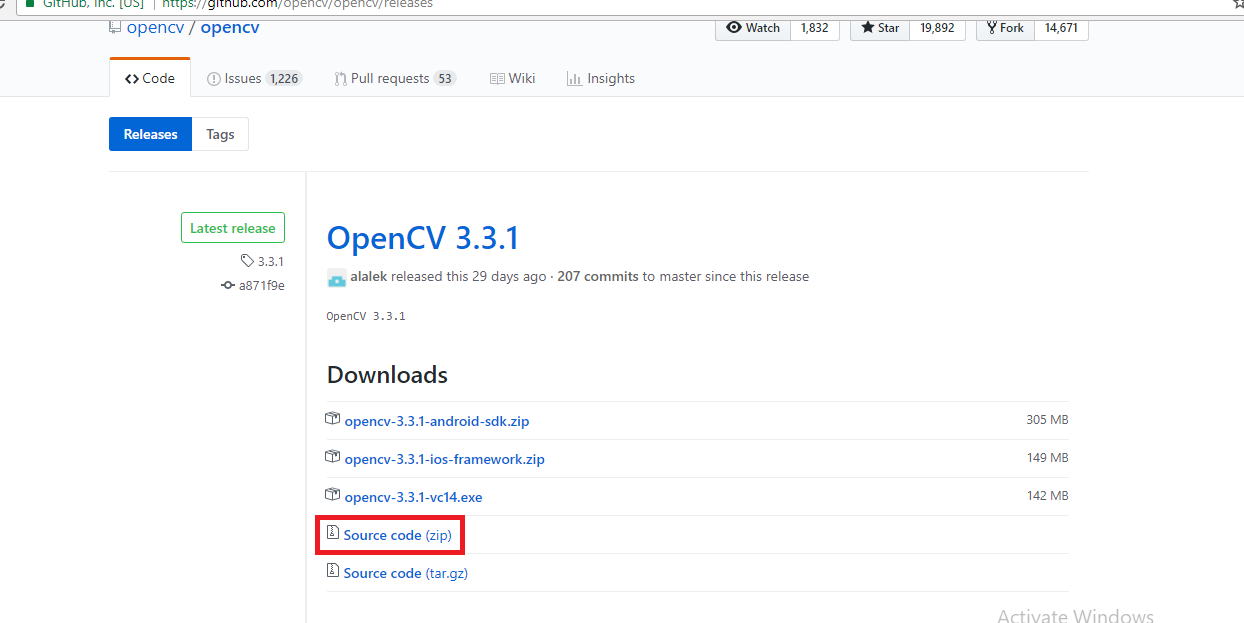
* Ha añadido Anaconda a la variable PATH en las variables de entorno
* Ha registrado Anaconda como el por defecto de python



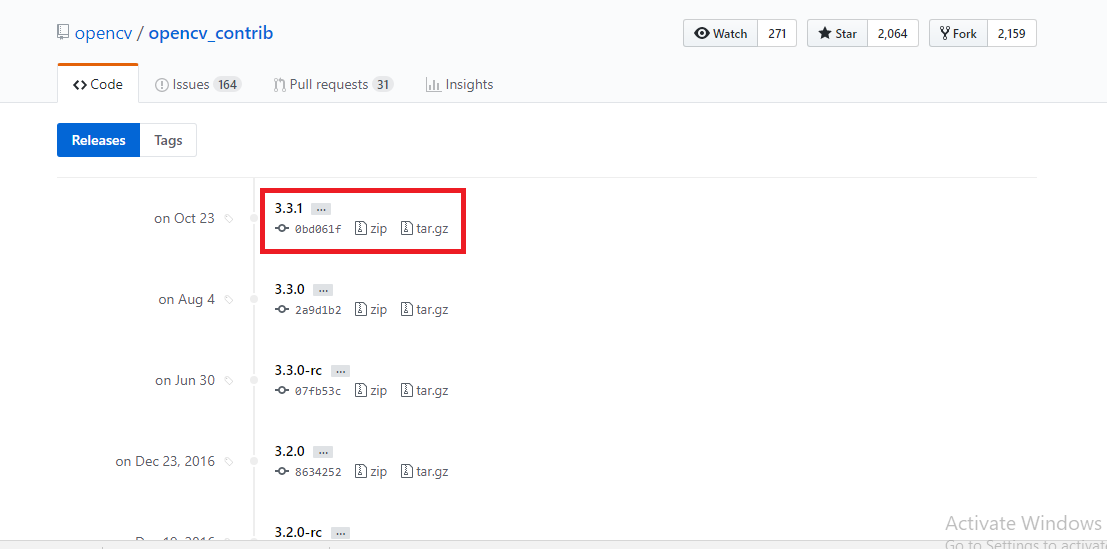
Paso 4 – Descargar e instalar opencv-3.4.0 y opencv\_contrib-3.4.0

Por favor note que las pantallas pueden variar y trate de hacer lo mismo si quiere instalar una nueva versión de opencv, esto es solo una referencia.

Vaya a <https://github.com/opencv/opencv/releases> y descargue opencv-3.4.0, carpeta de código fuente.



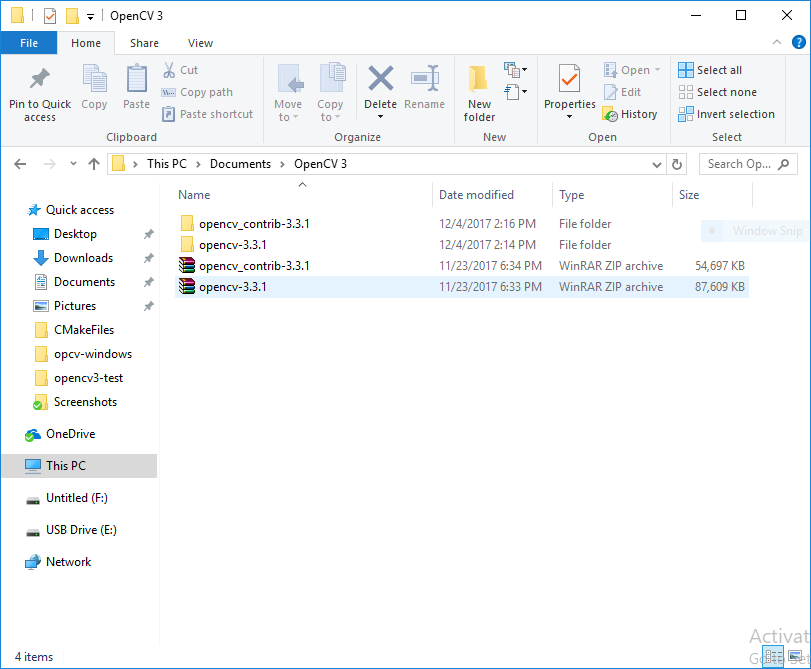
Vaya a <https://github.com/opencv/opencv_contrib/releases> y descargue opencv\_contrib-3.4.0.



Extraiga ambos zip. Aunque ud. Puede tener estos folders en cualquier parte del sistema, se recomienda mantener ambos en el mismo directorio. He posicionado estos folders en “Mis Documentos”.

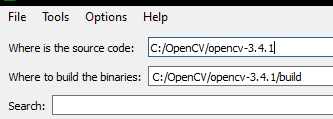
De aqui en Adelante nos referiremos al path de opencv como OPENCV\_PATH. En mi caso OPENCV\_PATH es C:/Users/pc/Documents/OpenCV3/opencv-3.4.0. Recuerde por favor de aquí en adelante OPENCV\_PATH que será mencionado multiples veces en el documento.

Dependiendo de la localización de su folder de opencv, este path será diferente.



Paso 5 – Generar un proyecto de Visual Studio utilizando CMake

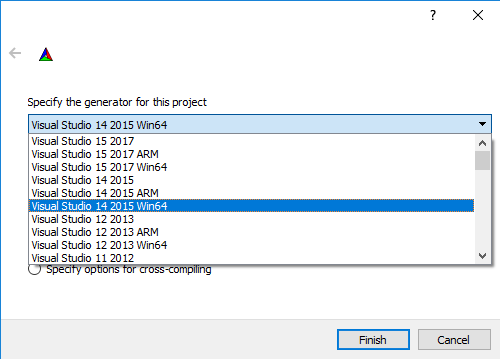
Correr CMake, el la caja “Where is the source code” escribir OPENCV\_PATH (que se menciono anteriormente) y en el path a construir seleccionamos OPENCV\_PATH/build.



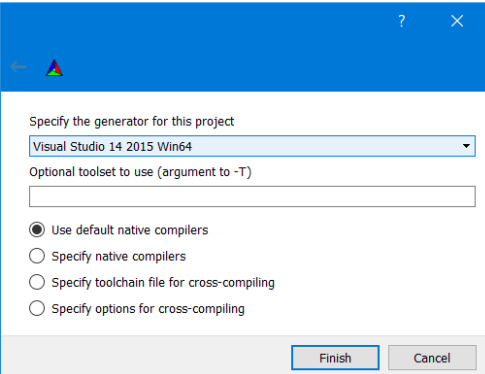
Ahora, hacer click en configure.

Se pedirá permiso para construir el folder buil, diga que sí.

Cuando sea el momento, seleccione el compilador, Visual Studio 14 2015 Win64 (en mi caso)



Presione finalizar y mantenga los parametros que se comentaron mantener.

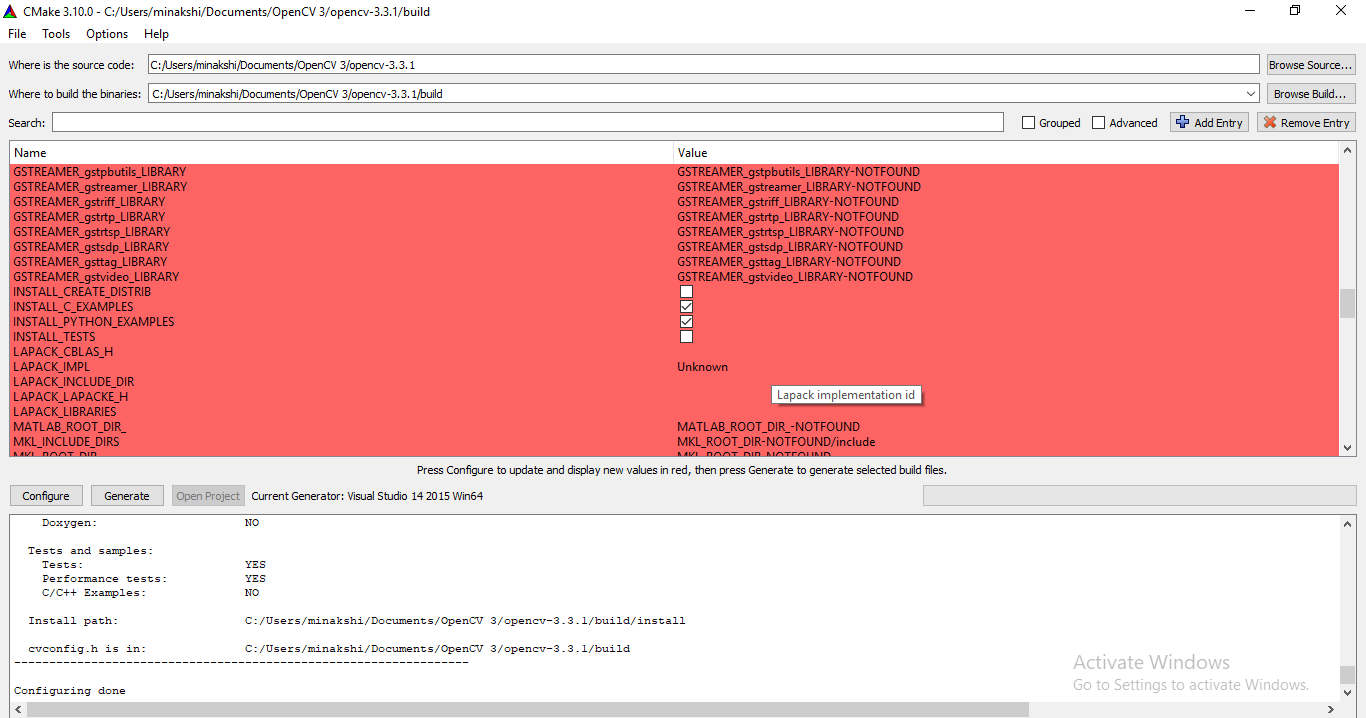


Presione Finalizar. Ahora CMake buscará en directorios del sistema y generará los archivos makefiles.

Paso 5.1 – Cambio adicional a CMake config

Realizaremos cambios en la configuración generada por CMake. Haga check e

* INSTALL\_EXAMPLES y INSTALL\_PYTHON\_EXAMPLES
* En la bandera OPENCV\_EXTRA\_MODULES\_PATH, de la ruta de los archivos contrib, por ejemplo C:/Users/control/Documents/OpenCV3/opencv\_contrib-3.4./modules
* Click en configure para aplicar los cambios



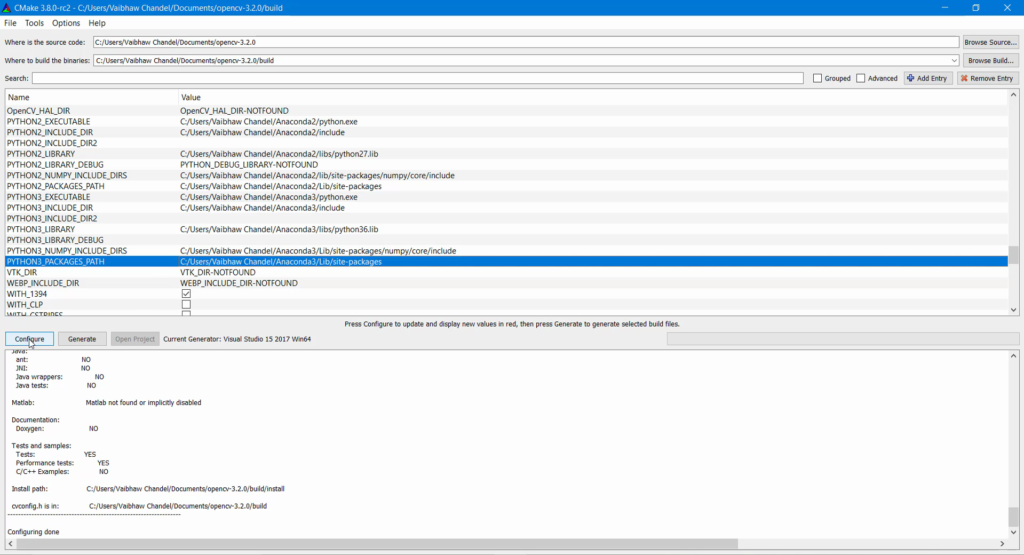
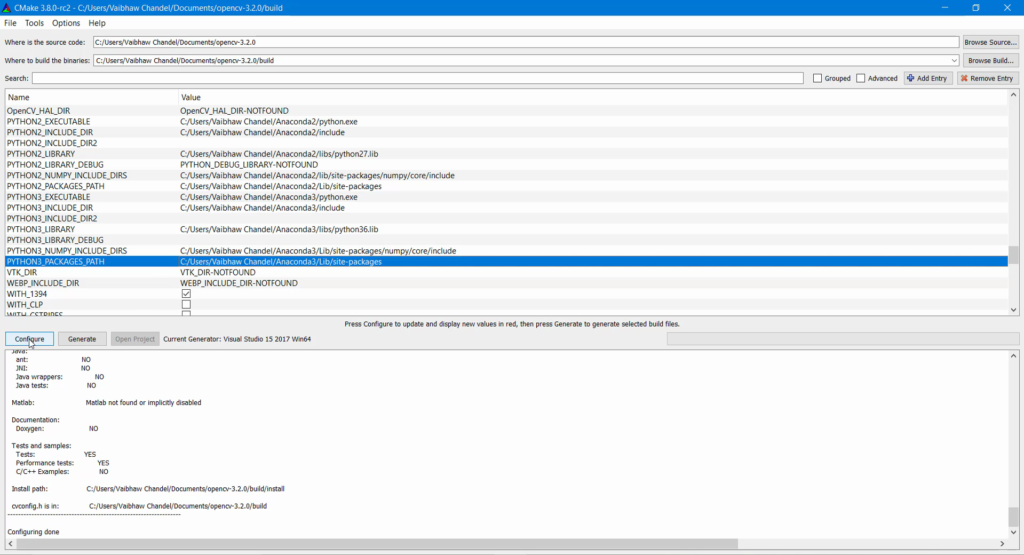
En Windows 10, opencv\_saliency falla alcompilar, así que debemos quitar el gancho a BUILD\_opencv\_saliency si lo encuentra puesto.

Haga click en configure nuevamente para incluir este cambio.

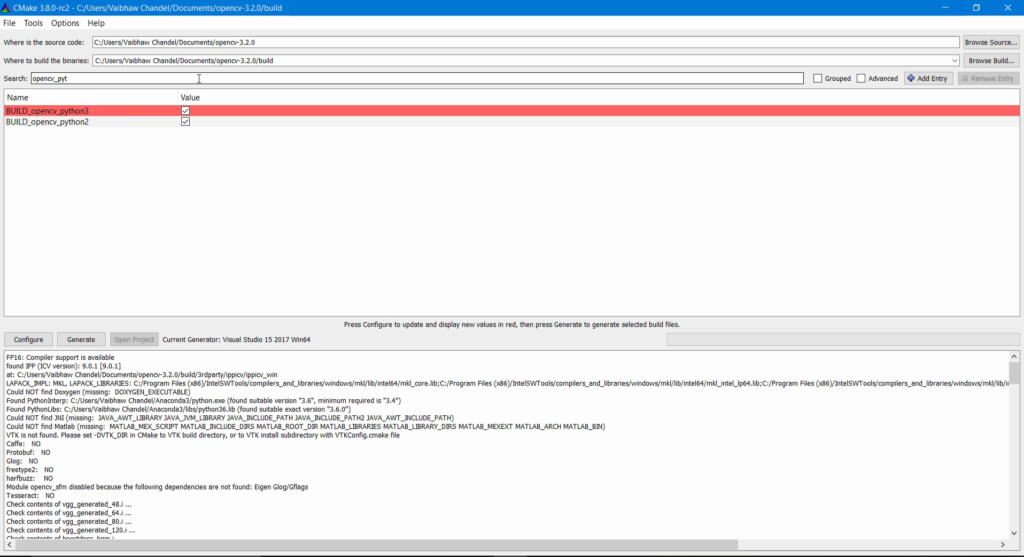
Paso 5.2 – Añadir la ruta de Python para Python2 y Python3

En esta sección generaremos los archivos binarios para ambos, Python2 y Python3. Si va a utilizar solamente Python3, debe obviar esta sección. Aquí nosotros siempre añadiremos los paths de ejecutables de python en los cuales CMake no ha de encontrar.

Si el path no es encontrado debe añadirse manualmente a los siguientes:

[](http://www.learnopencv.com/wp-content/uploads/2017/05/opencv_0055.png) [](http://www.learnopencv.com/wp-content/uploads/2017/05/opencv_0055.png)

Ahora hacer click en configurar nuevamente. Despues de que esta hecho en la barra de búsqueda de CMAKE ingresar opencv\_python y verificar que BUILD\_opencv\_pytho2 y BUILD\_opencv\_python3 estan con una marca.



### Paso 5.3 – Generar los archivos de build.

### Si CMake configuro todo correctamente sin errores debería de salir el mensaje “Configuring Done”. Hacer click en generate.

### NOTA: Cada vez que haga cambios debe repetir el proceso (marcar/no marcar las cajillas o cambiar rutas) de generar los archivos nuevamente.

### Paso 6 – Compilar OpenCV.

### Paso 6.1 – Compilar OpenCV en modo Release

### Abrir una ventana de comandos de windos e ir a OPENCV\_PATH/build, luego correr el siguiente comando:

cmake.exe --build . --config Release --target INSTALL

### Paso 6.2 – Compilar OpenCV en modo Debug

### Abrir CMake nuevamente como en el paso 5 y…

* Buscar “python” en la barra de búsqueda
* Deseleccionar INSTALL\_PYTHON\_EXAMPLES, BUILD\_opencv\_python3 y BUILD\_opencv\_python2
* Hacer click en configure
* Hacer click en generate

Ahora en una ventana de commandos ir a OPENCV\_PATH/build y corra el siguiente comando

cmake.exe --build . --config Debug --target INSTALL

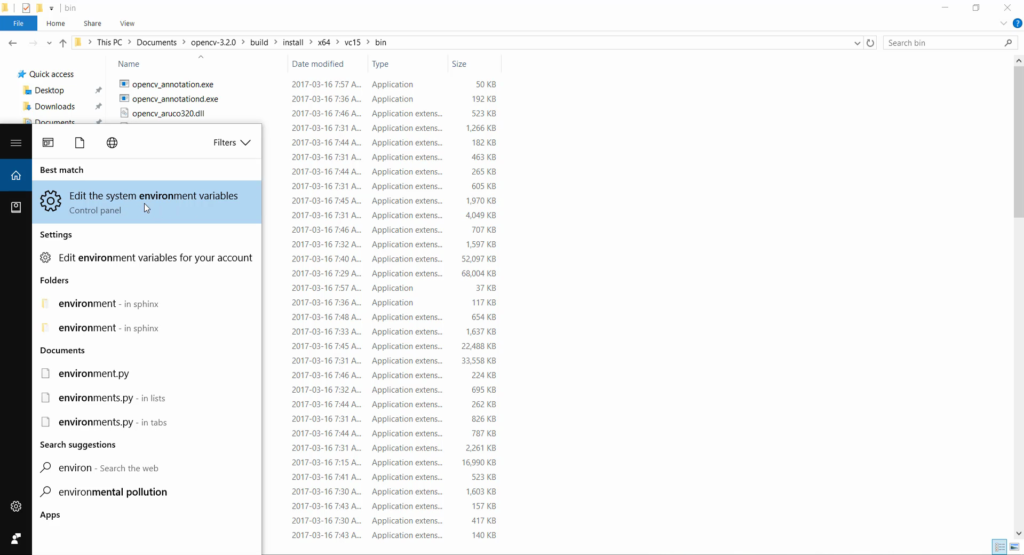
Instalado opencv procederemos a probar la instalación utilizando CMake

Paso 7 – Actualizar las variables de Entorno del Sistema

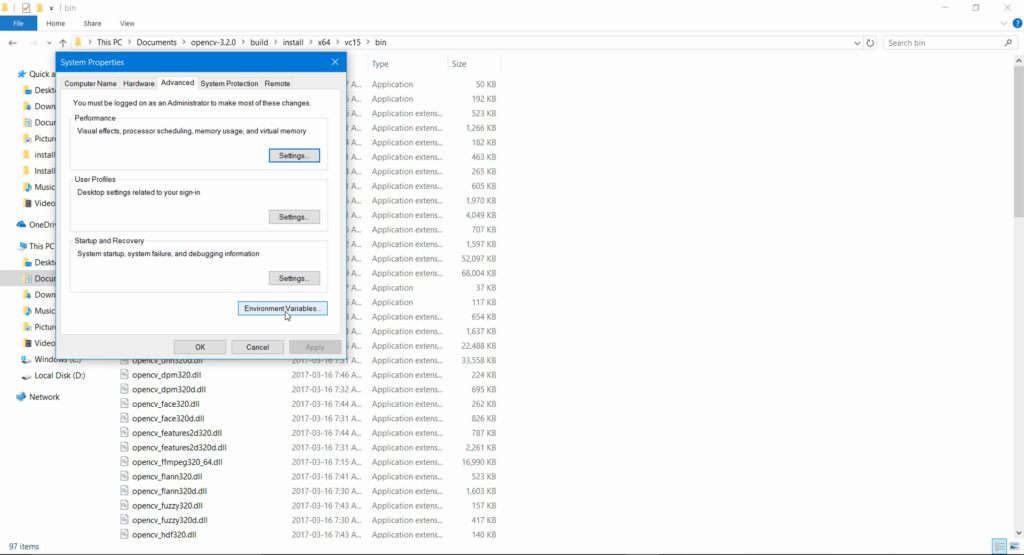
Paso 7.1 – Actualizar la variable de entorno – PATH

Primeramente añadir el dll de opencv a la variable de entorno PATH.

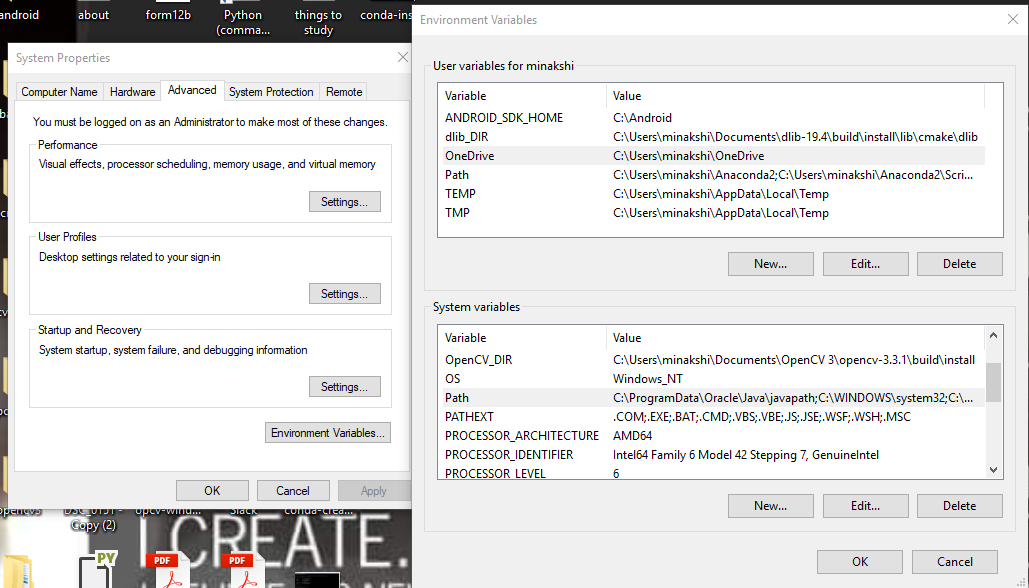
Presionar la tecla “Windows” y bucar “environment variables” o “variables de entorno”



Al hacer click en las variables de entorno en la ventana de propiedades del sistema



Debajo de variables del sistema seleccione Path y haga click en Edit

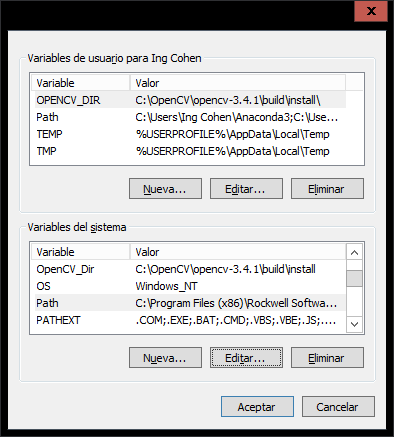


Hace click en Nuevo, y dar la ruta de OPENCV\_PATH\build\install\x64\vc15\bin por ejemplo, depende de la versión de visual studio instalada y hacer click en OK. Dependiendo de donde mantiene el folder de opencv y que versión de visual studio ha utilizado para compilar OpenCV, este path y versión de folder será diferente.

Por ejemplo, el mio será C:\Users\control\Documents\OpenCV 3\opencv-3.4.0\build\install\x64\vc15\bin

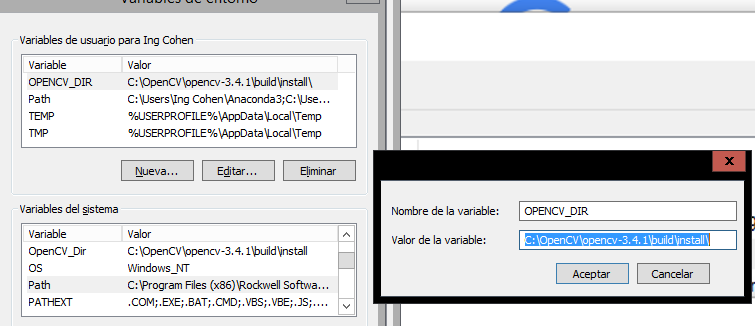
Ahora haga click en OK para salvar, no cierre la ventana para editar más variables.

Paso 7.2 – Actualizar las variables de entorno – OPENCV\_DIR



Hacer click en variables de usuario y debajo agregar un nombre de variable OPENCV\_DIR y como el valor de la variable OPENCV\_PATH\build\install.

Este directorio contiene el archivo “OpenCVConfig.cmake”. Este es usado por CMake para configurar OpenCV\_LIBS y OpenCV\_INCLUDE\_DIRS que son las variables que generan los proyectos.



Haga click en ok, salve y cierre las variables de entorno.

NOTA: Si ud tiene abierto la ventana de comandos de Power Shell antes de que estos valores se actualizaran deberá cerrar la ventana y repetir las acciones de nuevo.

Ajustando Dlib par C++

Si va a utilizar Dlib para C++, ud no tiene que instalar nada. Utilizaremos CMake para incluir el código fuente de Dlib con nuestro código de C++ y compilar Dlib con nuestro proyecto.

Paso 1 – Descargar Dlib

Se proveerá en la carpeta de proyecto en common, pero también se puede descargar del sitio web <http://www.dlib.net> y extraer a una carpeta del proyecto en un folder llamado common.

Paso 2 – Utilizando Dlib

Hay varias maneras de utilizar Dlib

* Build e Install como haremos con OpenCV
* Crear un proyecto utilizando CMake incluyendo el código fuente en el folder

La segunda opción es la mas sencilla y la recomendada por el creador de la librería Davis King.

El problema con la primera opción es que hay que definir directivas del pre-procesaodor y banderas de optimización para ud. En la sección final de este documento para Windows se verá como se ejecuta y valida toda la instalación anterior.

Ajustando para Python

Se debe instalar para python

Paso 1 – Instalar Dlib para python

Utilizaremos pip para instalar dlib ben python. Abrir la ventana de comandos y ejecutar el siguiente

pip install dlib

Comprobando la instalación de C++

* Abir el folder suministrado
* Abrir la temrinal y cambiar al directorio opencv/control/cpp, este debe contener un archivo CMakelists.txt y otro llamado facialLandmarkDetector.cpp
* Ejecute los siguientes comandos en la ventana

*# Create build directory to store all compiled binaries*

mkdir build

cd build

*# Configure for compilation*

cmake -G "Visual Studio 14 2015 Win64" ..

*# Compile the code*

cmake --build . --config Release

cd ..

* En el caso de que el comando falle…
  + Encontrar el directorio de instalación de OpenCV (OpenCV\_DIR), por ejemplo la localizada de OpenCVConfig.cmake.
  + Debe ser la misma dirección que en el paso obtenido en 7.2
  + Ingrese la ruta obtenida en el CMakelists.txt
* Correr el ejecutable

.\build\Release\facialLandmarkDetector

Se debe conseguir una salida como sigue a continuación:



Comprobando la instalación de Python

* Abir el folder suministrado
* Abrir la terminal en opencv/control/python
* Ejecutar el siguiente comando

python facialLandmarkDetector.py

Debe mostrarse la salida a continuación:



I.b – Instalación de OpenCVen MacOS

Paso 1 – Instalar XCode

* Instalar XCode del App Store
* Si XCode no está disponible para el SO utilizar
  + Encontrar la versión de XCode compatible para su SO en: <https://en.wikipedia.org/w/index.php?title=Xcode#Version_comparison_table>
  + Buscar XCode y bajar la versión compatible para su SO
* Instalar XCode
* Despues de instalar aceptar la licencia cuando salga

Paso 2 – Instalar Homebrew

Lanzar una terminal utilizando Launchpad. De aquí en adelante correr los comandos en la terminal

ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

brew update

*# Add Homebrew path to PATH variable*

echo "# Homebrew" >> ~/.bash\_profile

echo "export PATH=/usr/local/bin:$PATH" >> ~/.bash\_profile

source ~/.bash\_profile

*# Tap the science repo of homebrew*

*# This is not needed anymore since Homebrew recently moved many famous formulae from homebrew/science to homebrew/core*

brew tap homebrew/science

Paso 3 – Instalar Python2 y Python3

*# If installing python for the first time using Homebrew, else skip the 3 lines below and upgrade.*

brew install python python3

brew link python

brew link python3

*# NOTE : If you have python already installed using homebrew, it might ask you to upgrade. Upgrade the python using new homebrew formulae.*

brew upgrade python

brew upgrade python3

*# check whether Python using homebrew install correctly*

which python2 *# it should output /usr/local/bin/python2*

which python3 *# it should output /usr/local/bin/python3*

*# check Python versions*

python2 --version

python3 --version

Versiones requeridas de Python = 2.6, 2.7, 3.5, 3.6. Esto es requerido para verificar los site packages y será utilizada de aquí en adelante.

NOTA: Homebrew ha realizados algunos cambios en la nueva parte de python. Para viejas versiones se suele instlar python2 en /usr/local/bin/python. Ahora siga estas reglas.

* Instalar pytho2 en /usr/local/bin/python2
* Instalar python3 en /usr/local/bin/python3
* Python apuntará a /usr/bin/python. Esta es la distribución que viene en su SO y no esta instalada por Homebrew

Queremos utilizar Python instalado por Homebrew porque hace la administración de paquetes fácil. Para correr scripts ud debe correr comandos en python2 y python3 para Pytho 2 & 3 respectivamente. Si encuentra esto tedioso y quiere usar el comando python para correr python2, añadir la siguiente línea al ~/.bash\_profile.

*# Adding this line to end of .bash\_profile will make python command point to python2*

export PATH="/usr/local/opt/python/libexec/bin:$PATH"

Este paso es recomendado para tener la versión de python limpia instalada (p.e. para llamar en la ventana de comandos python <file>.py).

Paso 4 – Instalar Librerías de Python en el ambiente virtual de trabajo

Utilizaremos el ambiente virtual para instalar las librerías de python. Esto permitirá llevar seguimiento del proyecto y las variables de entorno globales por separado.

*# Install virtual environment*

pip2 install virtualenv virtualenvwrapper

echo "# Virtual Environment Wrapper"

echo "VIRTUALENVWRAPPER\_PYTHON=/usr/local/bin/python2" >> ~/.bash\_profile

echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.bash\_profile

source ~/.bash\_profile

*############ For Python 2 ############*

*# Create virtual environment*

mkvirtualenv cv-py2 -p python2

workon cv-py2

*# Install python libraries within this virtual environment*

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

*# Quit virtual environment*

deactivate

*############ For Python 3 ############*

*# Create virtual environment*

mkvirtualenv cv-py3 -p python3

workon cv-py3

*# Install python libraries within this virtual environment*

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

*# quit virtual environment*

deactivate

Paso 5 – Instalar OpenCV3

Paso 5.1 – Homebrew OpenCV

Compilar e instalar OpenCV. NOTA: Hombrew recientemente movio algunas cosas al core. Para instalaciones anteriores ud. Puede instalar OpenCV2 con el nombre opencv y OpenCV3 con opencv3.

Ahora han renombrado opencv3 para opencv y opencv para opencv2. Varias opciones de instalación como –with-qt –with-tbb etc han sido removidas de la instalación y el soporte de CUDA ha sido descartado.

Homebrew ha también hecho estricto la compilación e instalación de Python bindings para Python2 y Python3. Si ud no tiene Python3 instalado, homebrew lo instalará por ud mientras instala OpenCV.

*# Compile OpenCV 3 with Python2 and Python3 bindings*

brew install opencv

Paso 5.2 – Añadir OpenCV site-packages a la ruta global de site-packages

Cuando brew termina de compilar e instalar OpenCV3, correremos la actualización del path de los site-packages del directorio que contiene cv2.so al site-packages del directorio de Python. Dependiendo de la versión de python que uds posea (2.6/2.7 o 3.5/3.6) estas rutas serán diferentes.

*############ For Python 2 ############*

*# This is a single line command.*

echo /usr/local/opt/opencv/lib/python2.7/site-packages >> /usr/local/lib/python2.7/site-packages/opencv3.pth

*############ For Python 3 ############*

*# This is a single line command*

echo /usr/local/opt/opencv/lib/python3.6/site-packages >> /usr/local/lib/python3.6/site-packages/opencv3.pth

Paso 6 – Hacer un simlink de python en el ambiente virtual de trabajo

La ruta a OpenCV de python es diferente dependiendo de la versión de python que utilice. Verifique la ruta y el archivo en su máquina. Use este comando para encontrar la ruta de python en su máquina.

find /usr/local/opt/opencv/lib/ -name cv2\*.so

en su máquina usar la ruta exacta, por ejemplo.

*############ For Python 2 ############*

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

ln -s /usr/local/opt/opencv/lib/python2.7/site-packages/cv2.so cv2.so

*############ For Python 3 ############*

cd ~/.virtualenvs/cv-py3/lib/python3.6/site-packages/

ln -s /usr/local/opt/opencv/lib/python3.6/site-packages/cv2.cpython-36m-darwin.so cv2.so

Paso 7 – Probar la instalación de OpenCV

Ahora probaremos si openCV está instalado o no. Verificaremos el código de compilación y correremos unos ejemplos para las siguientes secciones.

*# activate virtual environment*

*############ For Python 2 ############*

workon cv-py2

*############ For Python 3 ############*

workon cv-py3

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

*# open ipython*

ipython

*# import cv2 and print version*

import cv2

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

*# If OpenCV3 is installed correctly, the above command should give output 3.4.0*

*# Press CTRL+D to exit ipython*

Instalar Dlib en MacOS

Ajustando Dlib para C++

Si ud va a utilizar Dlib para C++, ud no tiene que instalar nada. Utilizaremos CMake para incluir Dlib como fuente en su código de C++ y compilar Dlib con su proyecto.

Descargar Dlib

Dlib estará en el archivo opencv/common que se provee hoy. Ud puede también descargarlo del sitio web <http://www.dlib.net> , extraer la copia y hacer en su carpeta de proyecto <su\_ruta\_de\_proyecto>/common.

Utilizar la librería Dlib

Estas son varias maneras de utilizar Dlib en un proyecto:

* Instalar y hacer build de Dlib como hacimos en pasos anteriores
* Crear un proyecto usando CMake incluyendo el folder Dlib

La segunda opción es la mpas sencilla y la recomendada por el creador de Dlib, Davis King. El problema con la primera opción es que ud tiene un compilador y preprocedador que activar, veremos como el proyecto de C++ usa dlib para la siguiente sección.

Ajustando Dlib para python

Los usuarios de Python deben instalar Dlib para utilizar

Paso 1 – Instalar las librerías del sistema operativo

brew cask install xquartz

brew install gtk+3 boost

brew install boost-python --with-python3

### Paso 2 – Instalar Dlib

Usaremos pip para instalar Dlib

*############ For Python 2 ############*

*# activate virtual environment*

workon cv-py2

*# build and install Python module*

pip install dlib

*# exit from virtual environment*

deactivate

*############ For Python 3 ############*

*# activate virtual environment*

workon cv-py3

*# build and install Python module*

pip install dlib

*# exit from virtual environment*

deactivate

# Prueba del código de ejemplo de C++.

Estamos utilizando CMake para compilar archivos de C++. Puede ser complicaod al principio pero luego se volverá una rutina.

* Usar el folder opencv que se suministró
* Abrir una terminal y cambiar el directorio a: opencv/control/cpp. Debe contener una carpeta llamada facialLandmarkDetector.cpp.
* Ejecute el siguiente comando como sigue:

mkdir build

cd build

cmake ..

cmake --build . --config Release

cd ..

* En caso de que el comando de CMake falle:
  + Encuentre el directorio de instalación de OpenCV (OpenCV\_DIR) en la configuración del archivo cmake. Para realizar esto utilice el siguiente comando:
    - find /usr/local –name OpenCVConfig.cmake
    - Ud debe tener una salida similar a: /usr/local/Cellar/opencv/3.4.0/share/OpenCV/
  + Ingrese la ruta obtenida a nivel superior en el archivo CMakeLists.txt
  + Correr el ejecutable: ./build/facialLandmarkDetector
  + Debe tener la salida que sigue a continuación:



Prueba del código de Ejemplo en Python

* Utilizar el archivo suministrado.
* Abrir una terminal y cambiar a opencv/control/python
* Ejecute los siguientes comandos

*# activate virtual environment*

workon cv-py3

*# Execute sample code*

python facialLandmarkDetector.py

*# Exit virtual environment*

deactivate

Debe conseguir la salida que se muestra a continuación.

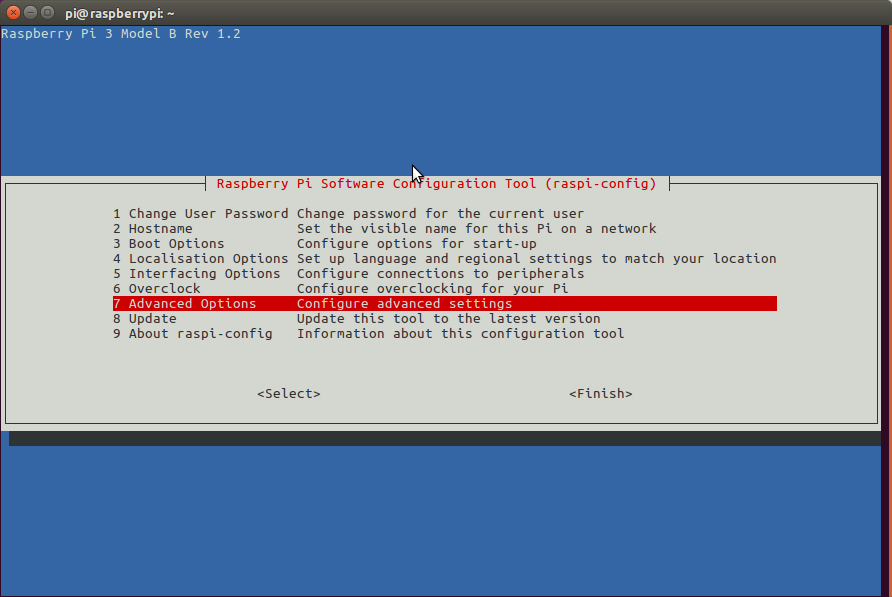


Seccion II – Instalar OpenCV para Raspberry Pi

Paso 1 – Expandir el sistema operativo

sudo raspi-config

Seleccionar opciones avanzadas

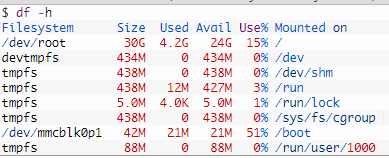


Seleccionar la opción de expandir el sistema operativo

Guardar, finalizar y reiniciar con…

sudo reboot –h now

Luego de reiniciar en una ventana de comandos escribir el comando df –h y podrá ver en /dev/root que se ha expandido el sistema, por ejemplo



Borre cosas que no utilizará como LibreOffice y Wolfram

sudo apt-get purge wolfram-engine

sudo apt-get purge libreoffice\*

sudo apt-get clean

sudo apt-get autoremove

Paso 2 – Instalar dependencias

OpenCV toma un proceso largo de compilación en el Raspberry Pi, asegurese de conectarlo a un UPS para que si hay fluctuaciones de voltaje o se pierde la energía no se pierda el trabajo realizado

Primeramente actualice los paquetes

sudo apt-get update && sudo apt-get upgrade

Instale cmake que nos ayudará a compilar opencv

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

Instalar otras dependencias de imagenes

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

Instalar otras dependencias de pero esta vez de video

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

sudo apt-get install libxvidcore-dev libx264-dev

Las siguientes dependencias son para utilizar higui de opencv

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

Importante es instalr las librerías de procesamiento de matrices

sudo apt-get install libatlas-base-dev gfortran

Consecuentemente instalamos python2 y python3

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

Instalar pre-requisitos de dlib

sudo apt-get update

sudo apt-get install build-essential cmake

sudo apt-get install libgtk-3-dev

sudo apt-get install libboost-all-dev

Verifque si pip está instalado

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

sudo python get-pip.py

Paso 3 – Descargar el código fuente de OpenCV

cd ~

wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.3.0.zip

unzip opencv.zip

Para usar cosas más avanzadas debemos utilizar las librerías de módulos

wget -O opencv\_contrib.zip https://github.com/Itseez/opencv\_contrib/archive/3.3.0.zip

$ unzip opencv\_contrib.zip

Si no se encuentra las librerías contrib, descargar de: <https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip>

Paso 4 – Python 2.7 vs Python 3

Antes de continuar debemos instalar pip

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

sudo python get-pip.py

sudo python3 get-pip.py

Instalar los ambientes virtuales de trabajo para el raspberry pi

sudo pip install virtualenv virtualenvwrapper

$ sudo rm -rf ~/.cache/pip

Actualizar el archivo ~./profile, añadir esto en el fondo

# virtualenv and virtualenvwrapper

export WORKON\_HOME=$HOME/.virtualenvs

export VIRTUALENVWRAPPER\_PYTHON=/usr/bin/python3

source /usr/local/bin/virtualenvwrapper.sh

pero también el paso anterior se puede realizar utilizando cat

echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile

echo "export WORKON\_HOME=$HOME/.virtualenvs" >> ~/.profile

echo "export VIRTUALENVWRAPPER\_PYTHON=/usr/bin/python3" >> ~/.profile

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

Pruebe esta instalación del ambiente virtual usando el comando source.

source ~/.profile

Creando ambientes virtuales, usar alguno de los dos

Para python2 crearemos el siguiente ambiente virtual

mkvirtualenv cv2 -p python2

Para python3 lo creamos de la misma manera

mkvirtualenv cv3 -p python3

Para probar el ambiente virtual ejecute lo siguiente, con uno de ambos es suficiente cv2 o cv3

source ~/.profile

workon cv2

deactivate

workon cv3

deactivate

Debe aparecer algo como (cv2) pi@raspberrypi si está dentro del ambiente, sino está fuera de él.

Instalar numpy

Ingrese al ambiente virtual

source ~/.profile

workon cv3

pip install numpy

Compilando e Instalando OpenCV

Ingresado al ambiente virtual ejecute el siguiente comando

cd ~/opencv-3.3.0/

mkdir build

cd build

cmake -D CMAKE\_BUILD\_TYPE=RELEASE \

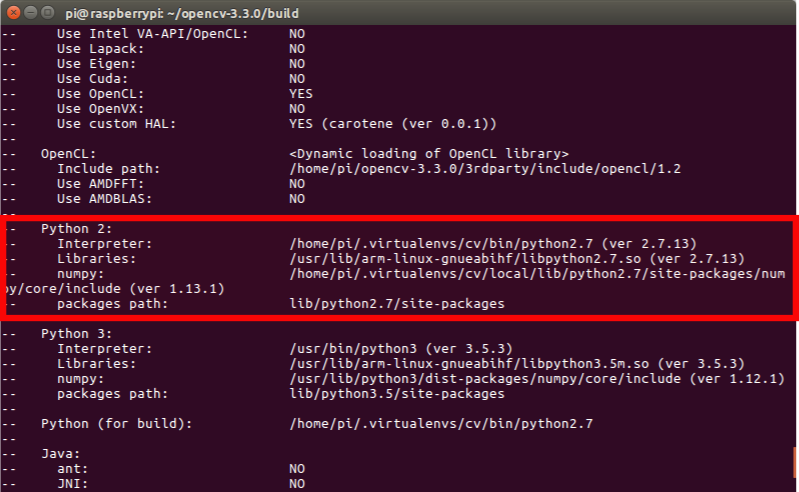
    -D CMAKE\_INSTALL\_PREFIX=/usr/local \

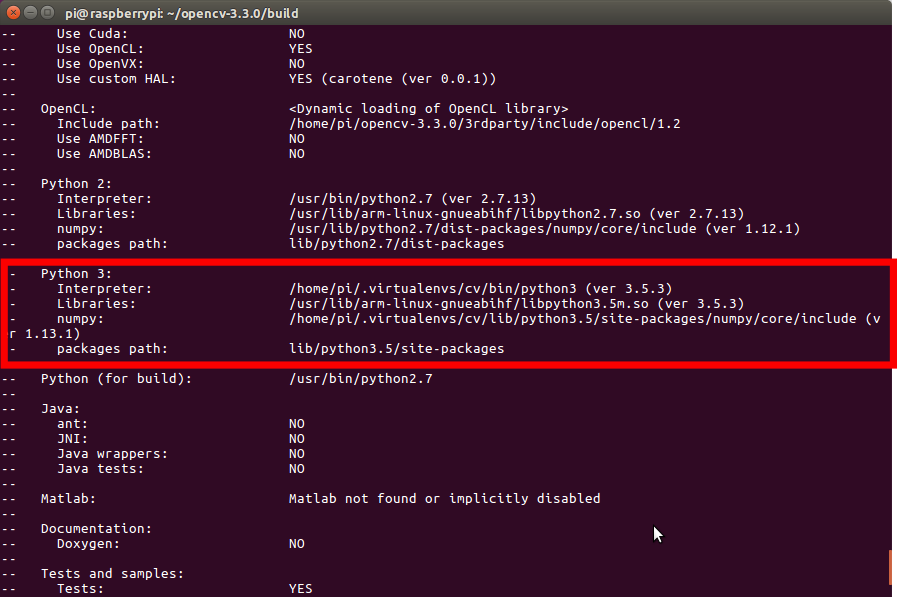
    -D INSTALL\_PYTHON\_EXAMPLES=ON \

    -D OPENCV\_EXTRA\_MODULES\_PATH=~/opencv\_contrib-3.3.0/modules \

    -D BUILD\_EXAMPLES=ON ..

Verifique la salida de CMake observando la salida de Python 2 y Python 3





De lo anterior observamos que se obtuvieron los site-packages correctos

Cofigure el swap space antes de compilar

Abir /etc/dphys-swapfile y editar

# set size to absolute value, leaving empty (default) then uses computed value

#   you most likely don't want this, unless you have an special disk situation

# CONF\_SWAPSIZE=100

CONF\_SWAPSIZE=1024

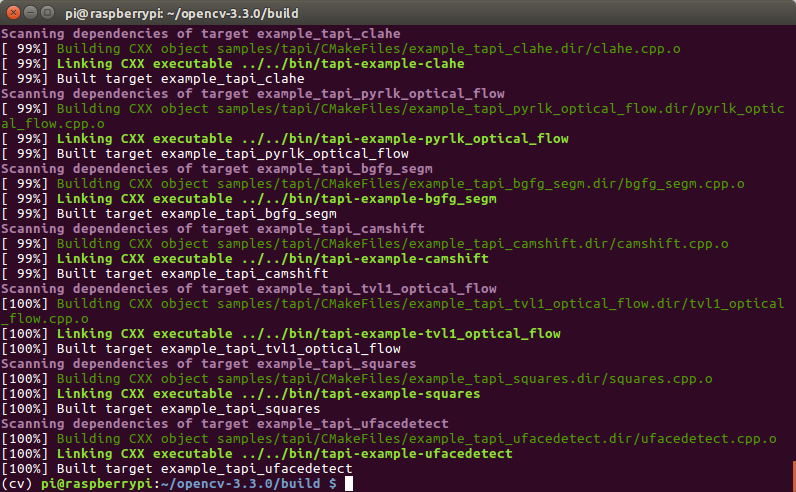
Note que swapsize está comentado, esto hara que todo fluya normalmente. Por consiguiente, grabe, salve lo anterior y ejecute el siguiente comando en la terminal.

sudo /etc/init.d/dphys-swapfile stop

$ sudo /etc/init.d/dphys-swapfile start

Finalmente compile opencv

make -j4



Terminada la compilación instale

sudo make install

sudo ldconfig

Para Python 2.7

Terminada la instalación podemos verificar que se instaló en:

/usr/local/lib/python2.7/site-pacakges

Para ello podemos correr el siguiente comando para verificar

ls -l /usr/local/lib/python2.7/site-packages/

total 1852

-rw-r--r-- 1 root staff 1895772 Mar 20 20:00 cv2.so

Lo ultimo es hacer un symlink a la variable de python de los site-packages

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

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

Para Python3

/usr/local/lib/python3.5/site-packages

ls -l /usr/local/lib/python3.5/site-packages/

total 1852

-rw-r--r-- 1 root staff 1895932 Mar 20 21:51 cv2.cpython-34m.so

cd /usr/local/lib/python3.5/site-packages/

sudo mv cv2.cpython-35m-arm-linux-gnueabihf.so cv2.so

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

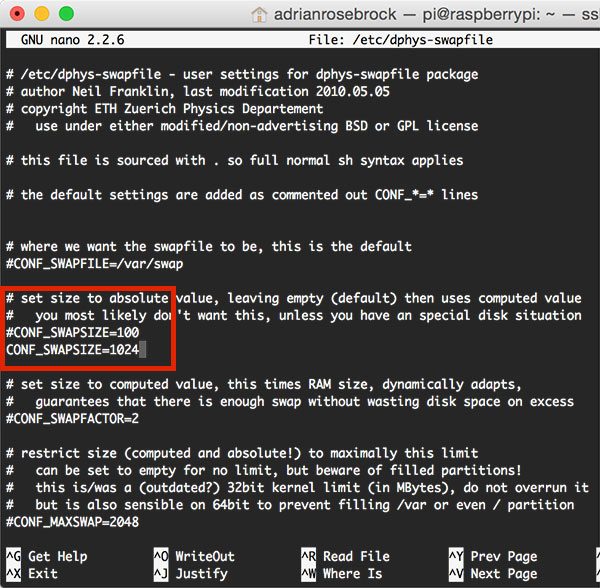
ln -s /usr/local/lib/python3.5/site-packages/cv2.so cv2.so

Instalando Dlib

Paso 1 – Actualizar archivo swap, opciones de arranque y memoria dividida

sudo nano /etc/dphys-swapfile

Verifque que el swap size está comentado



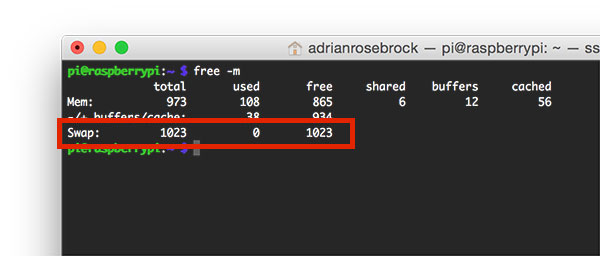
Correr el siguiente comando para actualizar el servicio

sudo /etc/init.d/dphys-swapfile stop

sudo /etc/init.d/dphys-swapfile start

Esto toma algunos minutos en ejecutarse, luego de finalizar libere espacio con

free –m



Cambiar opciones de arranque

Ejecutar

sudo raspi-config

Seleccionar

Boot Options => Desktop / CLI => Console Autologin

Actualizar la memoria dividida

En raspi-config regresar a AdvancedOptions => Memory Split

Cambiar el valor de 64 a 16

Reinicie el sistema

Ejecutar el siguiente commando para instalar las librerías que necesita Dlib, debe estar dentro de algún ambiente virtual

pip install numpy

pip install scipy

pip install scikit-image

Finalmente debe instalar dlib como sigue

pip install dlib

Probando la instalación de OpenCV

Ejecutar los siguientes comandos para verificar la versión

source ~/.profile

$ workon cv

$ python

>>> import cv2

>>> cv2.\_\_version\_\_

'3.3.0'

>>>

Probando la instalación de Dlib

Ejecutar el siguiente comando

$ 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 dlib

>>>

Paso 6 – Retornar algunas variables a su estado original

Para compilar y hacer que los paquetes funcionen deshabilitamos y bajamos opciones, ahora procederemos a habilitarlas

Abir /etc/dphys-swapfile y editar

# set size to absolute value, leaving empty (default) then uses computed value

#   you most likely don't want this, unless you have an special disk situation

CONF\_SWAPSIZE=100

CONF\_SWAPSIZE=1024

sudo raspi-config

Seleccionar

Boot Options => Desktop / CLI => Console Autologin

Actualizar la memoria dividida

En raspi-config regresar a AdvancedOptions => Memory Split

Cambiar el valor de 16 a 64

Probar OpenCV y Dlib

Copiar la carpeta suministrada al raspberry pi

Ejecutar el código de python

python facialLandmarkDetector.py

Debe tener la salida correspondiente listada debajo, probablemente el tamaño de la imagen tenga que cambiarse a uno de menor tamaño pues la pantalla del raspberry pi es pequeña.



O puede probar el siguiente código con una imagen que obtenga de internet de una cara

# import the necessary packages

from imutils import face\_utils

import dlib

import cv2

# initialize dlib's face detector (HOG-based) and then create

# the facial landmark predictor

p = "shape\_predictor\_68\_face\_landmarks.dat"

detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor(p)

# load the input image and convert it to grayscale

image = cv2.imread("example.jpg")

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# detect faces in the grayscale image

rects = detector(gray, 0)

# loop over the face detections

for (i, rect) in enumerate(rects):

# determine the facial landmarks for the face region, then

# convert the facial landmark (x, y)-coordinates to a NumPy

# array

shape = predictor(gray, rect)

shape = face\_utils.shape\_to\_np(shape)

# loop over the (x, y)-coordinates for the facial landmarks

# and draw them on the image

for (x, y) in shape:

cv2.circle(image, (x, y), 2, (0, 255, 0), -1)

# show the output image with the face detections + facial landmarks

cv2.imshow("Output", image)

cv2.waitKey(0)