Installation Required Packages

https://docs.opencv.org/trunk/d7/d9f/tutorial_linux_install.html

Required Packages

- GCC 4.4.x or later
- CMake 2.8.7 or higher
- Git
- GTK+2.x or higher including headers (libgtk2.0-dev)
- pkg-config
- Python 2.6 or later, Numpy 1.5 or later with developer packages (python-dev, python-numpy)
- ffmpeg or libav development packages: libavcodec-dev, libavformat-dev, libswscale-dev
- [optional] libtbb2 libtbb-dev
- [optional] libdc1394 2.x
- [optional] libjpeg-dev, libpng-dev, libtiff-dev, libjasper-dev, libdc1394-22-dev
- [optional] CUDA Toolkit 6.5 or higher

Getting OpenCV from Git Repository: Launch Git client and clone OpenCV repository. If you need modules from OpenCV contrib repository then clone it as well:

```
cd ~/<my_working_directory>
git clone https://github.com/opencv/opencv.git
git clone https://github.com/opencv/opencv_contrib.git
```

Steps Build From Source Distribution (1)

Building OpenCV from Source Using CMake

1. Create a temporary directory, denote as <cmake_build_dir> to put the generated Makefiles, project files as well the object files and output binaries, as:

cd ~/opencv
mkdir build
cd build

Note I choose /home/ubuntu/OpenCV/

```
ubuntu@ubuntu-ThinkPad-Yoga-14: ~/OpenCV
ubuntu@ubuntu-ThinkPad-Yoga-14:~/OpenCV$ ls
3rdparty  build  CMakeLists.txt   data  include  modules    README.md
apps      cmake  CONTRIBUTING.md  doc   LICENSE  platforms  samples
```

2. Configuring: cmake [<some optional parameters>] <path to the OpenCV source directory> For example

cmake -D CMAKE_BUILD_TYPE=Release -D CMAKE_INSTALL_PREFIX=/usr/local ..

```
ubuntu@ubuntu-ThinkPad-Yoga-14:/usr/local/bin$ ls
b43-fwcutter        opencv_createsamples  opencv_version
opencv_annotation   opencv_traincascade   opencv_visualisation
ubuntu@ubuntu-ThinkPad-Yoga-14:/usr/local/bin$ ./opencv_version
3.2.0
```

3. Description of some parameters

build type: CMAKE_BUILD_TYPE=Release\Debug

to build with modules from opencv_contrib set OPENCV_EXTRA_MODULES_PATH to <path to opencv_contrib/modules/>

set BUILD_DOCS for building documents

set BUILD_EXAMPLES to build all examples

Harry Li, Ph.D.

Steps Build From Source Distribution (2)

Building OpenCV from Source Using CMake

4. [optional] Building python. Set the following python parameters:
   PYTHON2(3)_EXECUTABLE = <path to python>
   PYTHON_INCLUDE_DIR = /usr/include/python<version>
   PYTHON_INCLUDE_DIR2 = /usr/include/x86_64-linux-gnu/python<version>
   PYTHON_LIBRARY = /usr/lib/x86_64-linux-gnu/libpython<version>.so
   PYTHON2(3)_NUMPY_INCLUDE_DIRS = /usr/lib/python<version>/dist-packages/numpy/core/include/
5. [optional] Building java.
6. Build. From build directory execute make, do this in several threads, for example
$make -j7 # runs 7 jobs in parallel
7. [optional] Building documents.

temporary directory <cmake_build_dir>

8. Install libraries, execute the following command from build directory
$sudo make install
9. [optional] Running tests by the required test data from OpenCV extra repository. For example
$git clone https://github.com/opencv/opencv_extra.git
set OPENCV_TEST_DATA_PATH environment variable to <path to opencv_extra/testdata>. Then execute tests from build directory, for example
   <cmake_build_dir>/bin/opencv_test_core

Harry Li, Ph.D.

Compile and Build with gcc and CMake

https://docs.opencv.org/trunk/db/df5/tutorial_linux_gcc_cmake.html

CMake is the easiest way with advantages: No need to change anything when porting between Linux and Windows, Can easily be combined with other tools by CMake(i.e. Qt, ITK and VTK)

1. Create a program using OpenCV

```
#include <stdio.h>
#include <opencv2/opencv.hpp>
using namespace cv;
int main(int argc, char** argv )
{
    if ( argc != 2 )
    {
        printf("usage: DisplayImage.out <Image_Path>\n");
        return -1;
    }
    Mat image;
    image = imread( argv[1], 1 );
    if ( !image.data )
    {
        printf("No image data \n");
        return -1;
    }
    namedWindow("Display Image", WINDOW_AUTOSIZE );
    imshow("Display Image", image);
    waitKey(0);
    return 0;
}
```

2 Create a CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)
project( DisplayImage )
find_package( OpenCV REQUIRED )
include_directories( ${OpenCV_INCLUDE_DIRS} )
add_executable( DisplayImage DisplayImage.cpp )
target_link_libraries( DisplayImage ${OpenCV_LIBS} )
```

Run DisplayImage

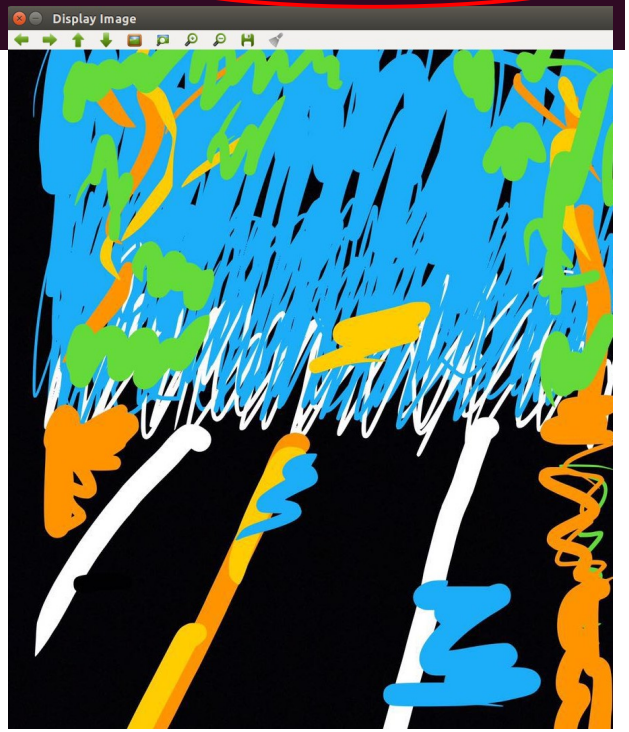
https://docs.opencv.org/trunk/db/df5/tutorial_linux_gcc_cmake.html

1

```
ubuntu@ubuntu-ThinkPad-Yoga-14:~/OpenCV/samples/cpp/harry-test/lec2-display$ cmake .
-- Found OpenCV: /usr/local (found version "3.2.0")
-- Configuring done
-- Generating done
-- Build files have been written to: /home/ubuntu/OpenCV/samples/cpp/harry-test/lec2-display
ubuntu@ubuntu-ThinkPad-Yoga-14:~/OpenCV/samples/cpp/harry-test/lec2-display$ make
Scanning dependencies of target DisplayImage
[100%] Building CXX object CMakeFiles/DisplayImage.dir/DisplayImage.cpp.o
Linking CXX executable DisplayImage
[100%] Built target DisplayImage
ubuntu@ubuntu-ThinkPad-Yoga-14:~/OpenCV/samples/cpp/harry-test/lec2-display$ ls
art-road1.jpg   CMakeFiles          CMakeLists.txt   DisplayImage      Makefile
CMakeCache.txt  cmake_install.cmake CMakeLists.txt~  DisplayImage.cpp
ubuntu@ubuntu-ThinkPad-Yoga-14:~/OpenCV/samples/cpp/harry-test/lec2-display$ ./DisplayImage art-road1.jpg
init done
opengl support available
```

2

3



1. $cmake .

2. $make

3. $./yourExecutable <_image_input>

Appendix: CMake

https://cmake.org/cmake-tutorial/

1. CMake a "common build system use cases"
2. The most basic project is an executable built from source code files. 2 lines CMakeLists.txt file can do the job as:

```
cmake_minimum_required (VERSION 2.6)
project (Tutorial)
add_executable(Tutorial tutorial.cxx)
```

Note lower case commands are used in this CMakeLists.txt file. Upper, lower, and mixed case commands are supported by CMake. The source code for tutorial.cxx computes the square root as follows:

```
// A simple program computes the square root
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main (int argc, char *argv[])
{
  if (argc < 2)
    {
    fprintf(stdout,"Usage: %s number\n",argv[0]);
    return 1;
    }
  double inputValue = atof(argv[1]);
  double outputValue = sqrt(inputValue);
  fprintf(stdout,"The square root of %g is %g\n",
          inputValue, outputValue);
  return 0;
}
```

Appendix: Use Cmake

1. Create your working directory, in my case, ~/OpenCV/samples/cpp/harry-test/lec1-Cmake$
2. Create “CmakeLists.txt” and the program to be compiled and built “Tutorial.cxx”

an old-style for C++ source files -- the plus signs turned 45 degrees

3. To start the process

$cmake .

```
ubuntu@ubuntu-ThinkPad-Yoga-14:~/OpenCV/samples/cpp/harry-test/lec1-Cmake$ cmake .
-- Configuring done
-- Generating done
-- Build files have been written to: /home/ubuntu/OpenCV/samples/cpp/harry-test/lec1-Cmake
```

Then, build the executable by
$make

```
ubuntu@ubuntu-ThinkPad-Yoga-14:~/OpenCV/samples/cpp/harry-test/lec1-Cmake$ make
Scanning dependencies of target Tutorial
[100%] Building CXX object CMakeFiles/Tutorial.dir/tutorial.cxx.o
Linking CXX executable Tutorial
[100%] Built target Tutorial
ubuntu@ubuntu-ThinkPad-Yoga-14:~/OpenCV/samples/cpp/harry-test/lec1-Cmake$ ls
CMakeCache.txt  cmake_install.cmake  CMakeLists.txt~  Tutorial      tutorial.cxx
CMakeFiles      CMakeLists.txt       Makefile         tutorial.cpp
```