Setting up OpenCV on macOS

1. Install homebrew

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

2. Install OpenCV

   ```
   brew install opencv
   ```

3. Update all installed packages

   ```
   brew update && brew upgrade && brew cleanup
   ```

4. Install pkg-config

   ```
   brew install pkg-config
   ```

pkg-config located at /usr/local/lib/pkgconfig/ where containing symlink to package, checking opencv4.pc exist or not.

If opencv4.pc does not exist, make a symlink:

- Put to ./bash_profile

  ```
  export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig/
  ```

- Make a symlink

  ```
  ln -s /usr/local/Cellar/opencv/4.0.1/lib/pkgconfig/opencv4.pc
  $PKG_CONFIG_PATH
  ```

5. Check OpenCV linker flags

   ```
   pkg-config --cflags --libs opencv4
   ```

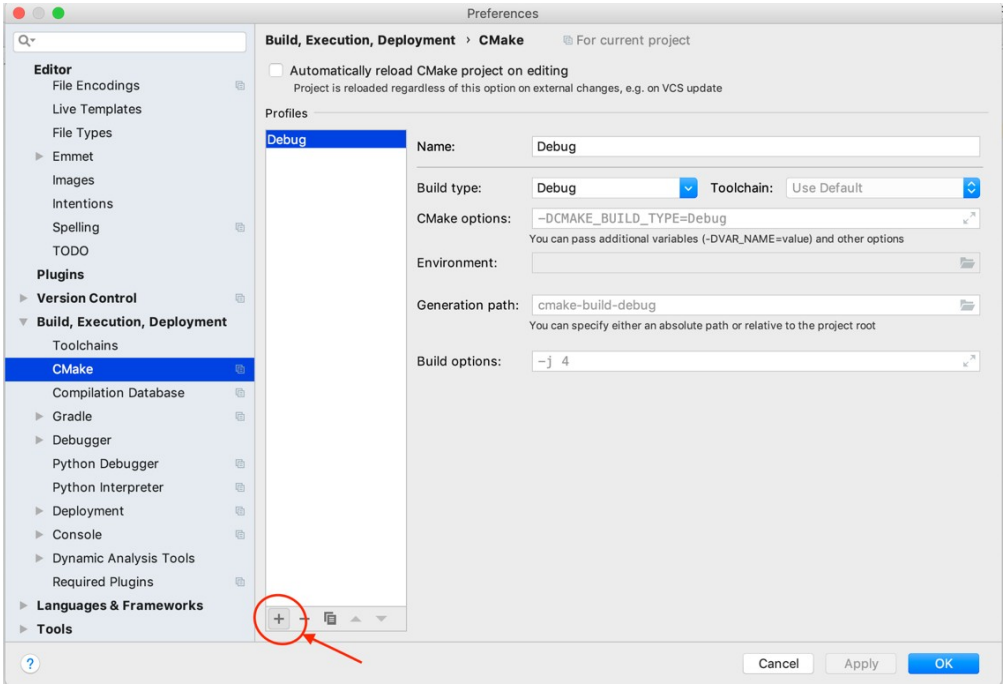
Should look like this:

```
-I/usr/local/Cellar/opencv/4.0.1/include/opencv4/opencv -
I/usr/local/Cellar/opencv/4.0.1/include/opencv4 -L/usr/local/Cellar/opencv/4.0.1/lib -lopencv_gapi -
lopencv_stitching -lopencv_aruco -lopencv_bgsegm -lopencv_bioinspired -lopencv_ccalib -
lopencv_dnn_objdetect -lopencv_dpm -lopencv_face -lopencv_fuzzy -lopencv_hfs -
lopencv_img_hash -lopencv_line_descriptor -lopencv_reg -lopencv_rgbd -lopencv_saliency -
lopencv_stereo -lopencv_structured_light -lopencv_phase_unwrapping -lopencv_superres -
lopencv_optflow -lopencv_surface_matching -lopencv_tracking -lopencv_datasets -lopencv_dnn -
lopencv_plot -lopencv_videostab -lopencv_video -lopencv_xfeatures2d -lopencv_shape -lopencv_ml
-lopencv_ximgproc -lopencv_xobjdetect -lopencv_objdetect -lopencv_calib3d -lopencv_features2d -
lopencv_highgui -lopencv_videoio -lopencv_imgcodecs -lopencv_flann -lopencv_xphoto -lopencv_photo
-lopencv_imgproc -lopencv_core
```

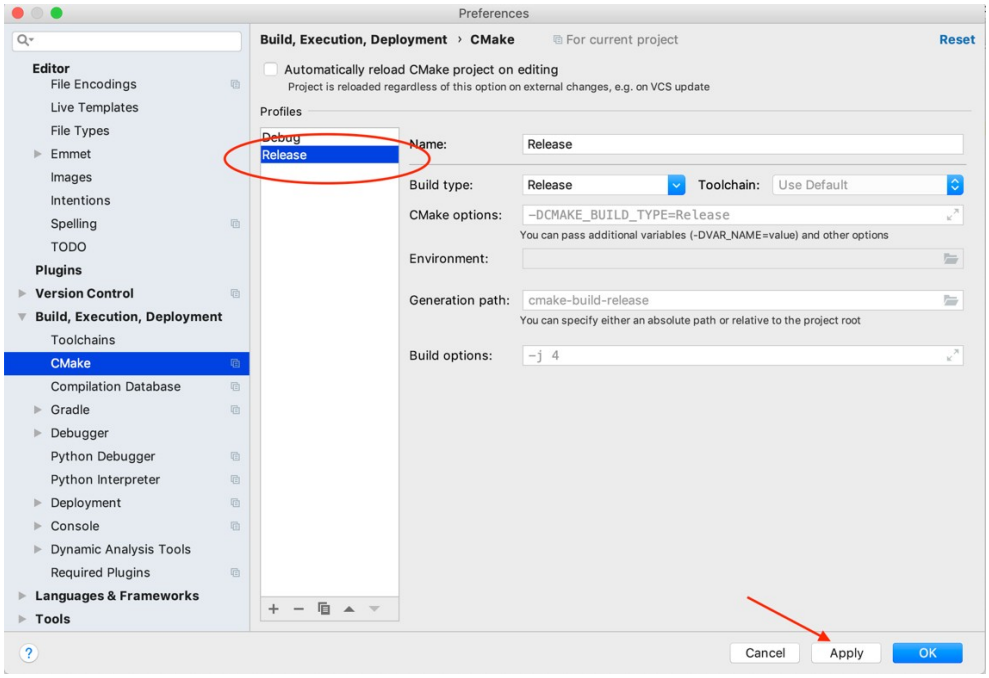
6. Setting up OpenCV on CLion IDE

- Change the configuration of CMake from debug to release, as following:

Toolbar -> CLion -> Preferences -> Build, Execution, Deployment -> CMake

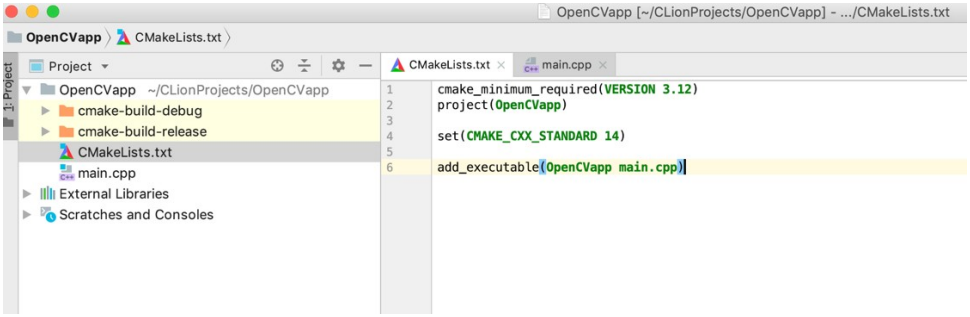


After clicking on + it should look like this:



Apply changes.

- Link the OpenCV library by configuring the CMakeLists file



The content of this file should be:

```
# cmake needs this line
cmake_minimum_required(VERSION 3.17)
# Enable C++11
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED TRUE)
if(NOT CMAKE_BUILD_TYPE AND NOT (MSVC_IDE OR XCODE))
   set(CMAKE_BUILD_TYPE Release CACHE STRING "Build type, one of: Release, Debug"
       FORCE)
endif()
# Set target name, change with your favorite name
project(OpenCVapp CXX)
message( STATUS "Build type: ${CMAKE_BUILD_TYPE}" )
message( STATUS "Configuration types: ${CMAKE_CONFIGURATION_TYPES}" )
# Detect 64 or 32 bit
if (CMAKE_SIZEOF_VOID_P EQUAL 8)
   # 64-bit project
   SET(64_BIT_OS TRUE)
   message( STATUS "A 64-bit OS detected" )
else()
   SET(64_BIT_OS FALSE)
   message( STATUS "A 32-bit OS detected" )
endif()
# Find OpenCV, you may need to set OpenCV_DIR variable
# to the absolute path to the directory containing OpenCVConfig.cmake file
# via the command line or GUI
if(WIN32)
   set(OpenCV_DIR "D:/_Program Files/opencv/build/x64/vc15/lib")
endif()
if(APPLE)
   set(OpenCV_DIR "/usr/local/Cellar/opencv/4.5.1_2/include/opencv4")
endif()
find_package(OpenCV REQUIRED)
# If the package has been found, several variables will
# be set, you can find the full list with descriptions
# in the OpenCVConfig.cmake file.
# Print some message showing some of them
message(STATUS "OpenCV library status:")
message(STATUS " config: ${OpenCV_DIR}")
message(STATUS " version: ${OpenCV_VERSION}")
message(STATUS " libraries: ${OpenCV_LIBS}")
message(STATUS " include path: ${OpenCV_INCLUDE_DIRS}")
file(GLOB srcs *.cpp *.c)
file(GLOB hdrs *.hpp *.h)
include_directories("${CMAKE_CURRENT_LIST_DIR}")
# Since there are a lot of examples I'm going to use a macro to simplify this
# CMakeLists.txt file. However, usually you will create only one executable in
# your cmake projects and use the syntax shown above.
macro(add_example name)
```

```
    add_executable(${name} ${name}.cpp)
    target_link_libraries(${name} ${OpenCV_LIBS} )
endmacro()
# if an example requires GUI, call this macro to check DLIB_NO_GUI_SUPPORT to include or exclude
macro(add_gui_example name)
    if (DLIB_NO_GUI_SUPPORT)
        message("No GUI support, so we won't build the ${name} example.")
    else()
        add_example(${name})
    endif()
endmacro()
add_example(main)
```

The only changes that need to be applied are on the lines 10 and 59. On line 10 should be the name of the project and on the 59th line should be all the files that need to be compiled (usually, all the .cpp files). Obviously, `cmake_minimum_required(VERSION 3.17)` has to be the current version. If you put a new version that you have not updated yet, at runtime you will get an error (the error will also output your version of cmake). Also you have to pay attention at line 30 such that the opencv version is your version.

Project example:

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

using namespace std;
using namespace cv;

int main() {
    Mat img = imread("/Users/mirunabetianu/CLionProjects/OpenCVapp/saturn.bmp",
    IMREAD_GRAYSCALE);
    Mat outImg(img.rows, img.cols, CV_8UC1);
    cout<<img.rows;

    for(int i = 0; i < img.rows; i++)
    {
        for(int j = 0; j < img.cols; j++)
        {
            outImg.at<uchar>(i,j) = 255 - img.at<uchar>(i,j);
        }
    }

    imshow("input image", img);
    imshow("output image", outImg);
    waitKey(0);
    return 0;
}
```

This program computes the negative of the input image, as below:

Note: the 1st parameter of the *imread* function needs to receive the absolute path of the source image.