Manual
2021/02/20

Hello world OpenCV with VS Code

Installation OpenCV in MacOS

Installing CMake

CMake is a cross platform build tool popular among native C/C++ developers. I use CMake in this tutorial to build and install OpenCV for C++ as well as run the demo project. CMake can be easily installed using brew as follows.

```
# brew install cmake
```

Installing OpenCV from Source with CMake

```
# mkdir opencv
# cd opencv
# git clone https://github.com/opencv/opencv.git
# cd opencv
# git checkout tags/4.2.0
# cd ..
# git clone https://github.com/opencv/
# cd opencv_contrib
# git checkout tags/4.2.0
# cd ..
```

I should now be in the top level opencv directory that contains the two aforementioned repositories.
Next I create one more directories named **build_opencv** for building the sources in.

```
# mkdir install build_opencv
# ls -l
total 0
drwxr-xr-x   2 thainq   64 Mar 29 21:45 build_opencv
drwxr-xr-x  21 thainq  672 Mar 29 21:35 opencv
drwxr-xr-x  13 thainq  416 Mar 29 21:36 opencv_contrib
```

Next I change directories into the build_opencv directory and configure CMake as shown below.

Installation OpenCV in MacOS

```
# cd build_opencv
# cmake -D CMAKE_BUILD_TYPE=RELEASE \
      -D OPENCV_EXTRA_MODULES_PATH=../opencv_contrib/modules \
      -D BUILD_DOCS=ON
      -D BUILD_EXAMPLES=ON ../opencv
```

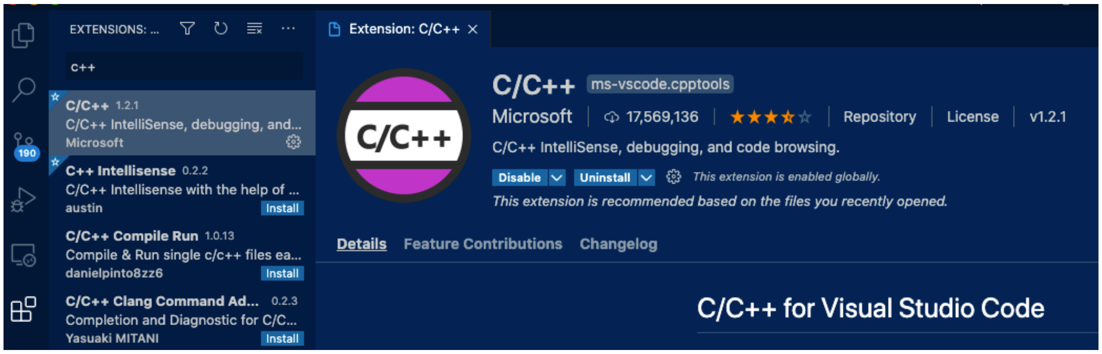
The last steps to install is to kick off the multithreaded build to compile the project followed by installing it into the install directory as seen below.

```
# make -j 7 # runs 7 jobs in parallel
# make install
```

Installation CMake in VS Code

Install C/C++ extension

C++ extension for VS Code. Install the C/C++ extension by searching for 'c++' in the Extensions view



C/C++ 1.2.1
C/C++ IntelliSense, debugging, and...
Microsoft

C++ Intellisense 0.2.2
C/C++ Intellisense with the help of ...
austin

C/C++ Compile Run 1.0.13
Compile & Run single c/c++ files ea...
danielpinto8zz6

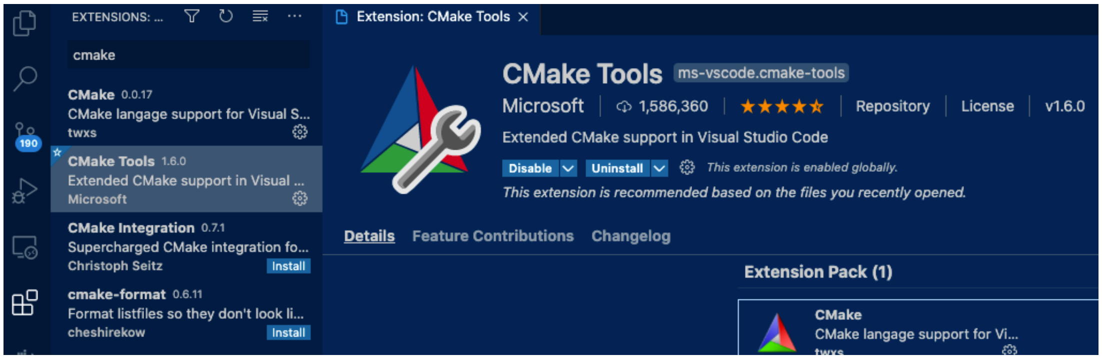
C/C++ Clang Command Ad... 0.2.3
Completion and Diagnostic for C/C...
Yasuaki MITANI

C/C++ ms-vscode.cpptools
Microsoft | 17,569,136 | ★★★★☆ | Repository | License | v1.2.1
C/C++ IntelliSense, debugging, and code browsing.
This extension is enabled globally.
This extension is recommended based on the files you recently opened.

C/C++ for Visual Studio Code

Install CMake plugin for VS code

C++ extension for VS Code. Install the C/C++ extension by searching for 'cmake' in the Extensions view



CMake 0.0.17
CMake langage support for Visual S...
twxs

CMake Tools 1.6.0
Extended CMake support in Visual ...
Microsoft

CMake Integration 0.7.1
Supercharged CMake integration fo...
Christoph Seitz

cmake-format 0.6.11
Format listfiles so they don't look li...
cheshirekow

CMake Tools ms-vscode.cmake-tools
Microsoft | 1,586,360 | ★★★★★ | Repository | License | v1.6.0
Extended CMake support in Visual Studio Code
This extension is enabled globally.
This extension is recommended based on the files you recently opened.

Extension Pack (1)

CMake
CMake langage support for Vi...
twxs

Ensure that CMake is installed

The VS Code CMake Tools extension does its work by using CMake installed on your system. For best results, use CMake version 3.15 or greater
See if CMake is already installed on your system. Open terminal window and enter the following command:

```
# cmake –version
cmake version 3.19.2
```

Ensure that development tools are installed

```
# gcc -v
Apple clang version 12.0.0 (clang-1200.0.32.27)
Target: x86_64-apple-darwin19.6.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
```

Hello World

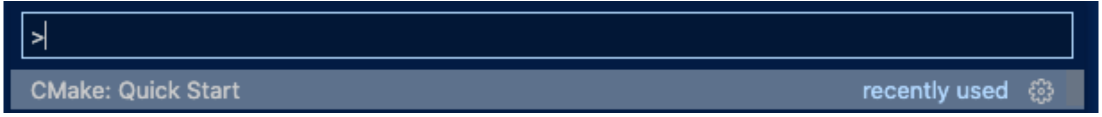
Open CV

Create a folder for a new project

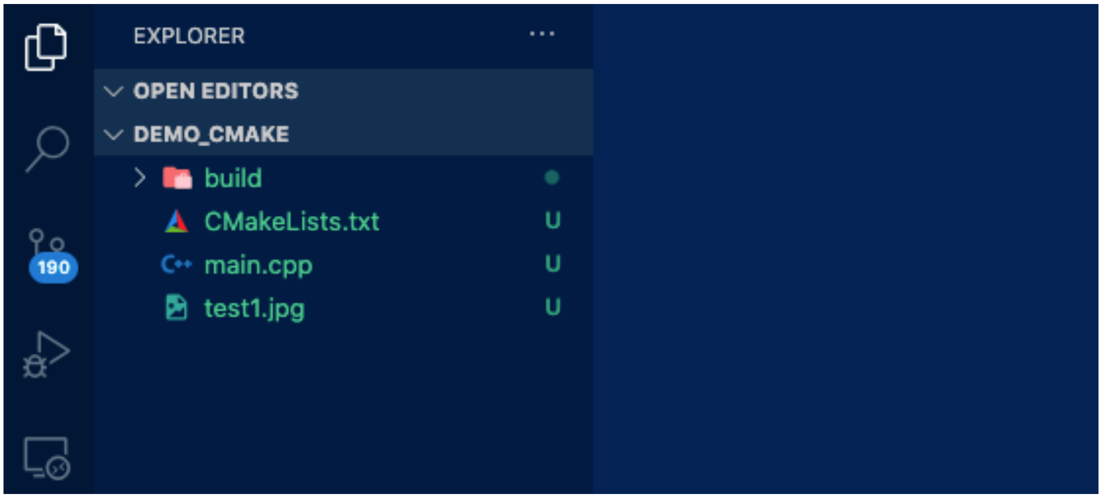
```
# mkdir hello-world-opencv
# cd hello-world-opencv
# code .
```

Create a CMake hello world project

The CMake tools extension can create the fields for basic CMake porjetc for you. Open the Command Palette and run the **CMake: Quick Start** command



Enter a project name: hello-world-opencv
Next, select Executable as the project type to create a basic source file (main.cpp) that includes a basic main() function



Update CMakeLists.txt with following instruction

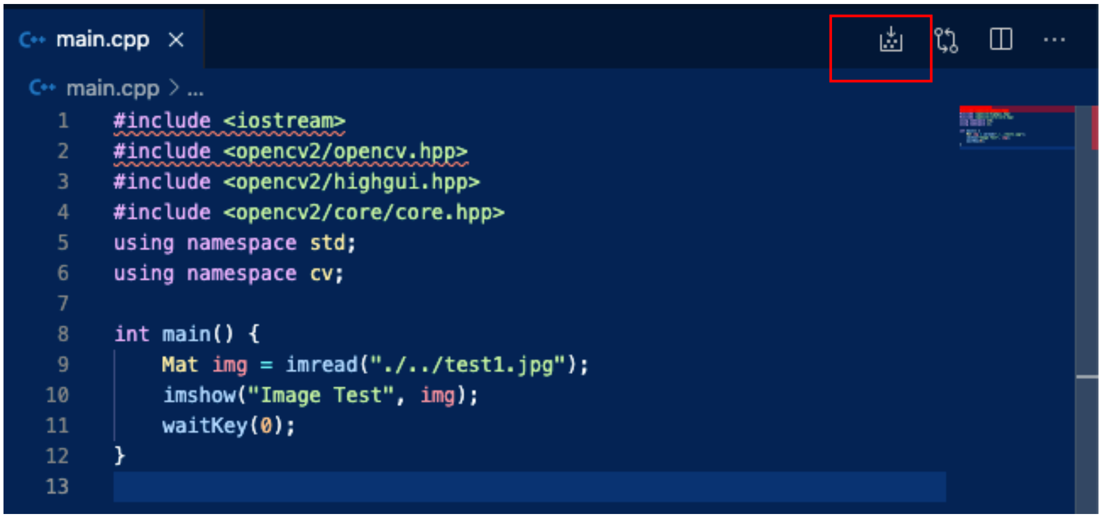
```
cmake_minimum_required(VERSION 3.0.0)
project(hello-world-opencv VERSION 0.1.0)

include(CTest)
enable_testing()

add_executable(demo_open_cv main.cpp)

set(CPACK_PROJECT_NAME ${PROJECT_NAME})
set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})

# Tell compiler to use C++ 14 features which is needed because
# Clang version is often behind in the XCode installation
set(CMAKE_CXX_STANDARD 14)

# set OpenCV_DIR variable equal to the path to the cmake
# files within the previously installed opencv program
```

```
set(OpenCV_DIR /usr/local/include/opencv4)

# configure the necessary common CMake environment variables
# needed to include and link the OpenCV program into this
# demo project, namely OpenCV_INCLUDE_DIRS and OpenCV_LIBS
find_package( OpenCV REQUIRED )

# tell it to link the executable target against OpenCV
target_link_libraries(demo_open_cv ${OpenCV_LIBS} )
include(CPack)
```

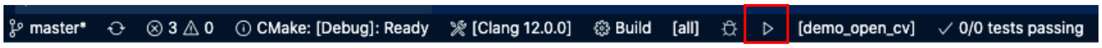
Update file main.cpp with following code

```
#include <iostream>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/core/core.hpp>
using namespace std;
using namespace cv;

int main() {
    Mat img = imread("./../test1.jpg");
    imshow("Image Test", img);
    waitKey(0);
}
```

Buid & Run



```
#include <iostream>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/core/core.hpp>
using namespace std;
using namespace cv;

int main() {
    Mat img = imread("./../test1.jpg");
    imshow("Image Test", img);
    waitKey(0);
}
```



Settings for vscode

Under folder .vscode
Making a new file named: c_cpp_properties.json with content as below

```
{
    "configurations": [
        {
            "name": "Mac",
            "includePath": [
                "${workspaceFolder}/*",
                "/usr/local/include/opencv4/"
            ],
            "defines": [],
            "macFrameworkPath": [

"/Library/Developer/CommandLineTools/SDKs/MacOSX10.15.sdk/System/Library/Frameworks"
            ],
            "compilerPath": "/usr/bin/clang",
            "cStandard": "c11",
            "cppStandard": "c++17",
            "intelliSenseMode": "clang-x64",
            "configurationProvider": "ms-vscode.cmake-tools"
        }
    ],
    "version": 4
}
```