

C/C++ programming with Qt Creator and OpenCV versions from package manager

Preparation of the computer (Windows)

- Please preferably use the [.doc](#) version of this document since copy-paste from many [.pdf](#) readers miss/add/change some characters, typically hyphen, space, newline, quotation marks, etc.
- Install Chocolatey package manager: <https://chocolatey.org/install> and then install CMake with `choco install -y cmake.install --installargs 'ADD_CMAKE_TO_PATH=System'`, make with `choco install -y make`, Qt with `choco install -y qt6-base-dev --version=6.4.2.202408081` and `choco install -y qtcreator2`. You might also want to prevent further updates with `choco pin add -n qt6-base-dev`, `choco pin add -n qtcreator`.
- Install OpenCV package with `choco install -y libopencv-dev --version=4.6.0.20240808`. You might also want to prevent further updates with `choco pin add -n libopencv-dev`.
- Restart.

Preparation of the computer (Ubuntu)

- Install Qt with `sudo apt -y install qtcreator qt6-base-dev libgl1-mesa-dev build-essential cmake` (since Ubuntu 22.04) or `sudo apt -y install qtcreator qt5-default build-essential cmake`, additional/different steps might be necessary depending on the specific versions (and the provided `.pro` might need to be tweaked).
- For OpenCV, see https://www.ensta-bretagne.fr/lebars/Share/setup_opencv_Ubuntu.pdf.

Tricks/common problems Qt Creator

- Note that when you run your application from Qt Creator, the **working directory of the application** is set by default to some build folder, check project **Run Settings** to change that.
- **Create projects only on local disks** (e.g. in `C:\TEMP\OPENCV\`). Network disks might not be fully supported.
- **Delete** the generated files and folders `build`, `.qtcreator`, `*.pro.user`, `CMakeLists.txt.user`, `*-build-desktop` when moving your project or when the project behavior looks inconsistent (e.g. a wrong version of your program in another path is launched), and reopen the project to force Qt Creator to regenerate them. Usually, only the source files (`.c`, `.cpp`, `.h`) and `.pro` file (also `.ui` if using Qt specific GUI functions, `CMakeLists.txt` if using CMake, etc.) are required, as well as optional data that might be specific to your application (e.g. images to process...).

¹ A compatible version of MinGW GCC should be automatically installed as dependency.

² In case the installation fails, try to uninstall using `choco uninstall -y qtcreator` and if needed delete `C:\tools\qtcreator` before trying to install again, possibly another version.

- If you do not see inside **Qt Creator Application Output** window the output of **printf()** or other functions that write on **stdout**, check that you have **CONFIG += console** in your **.pro**. Check also **Projects > Run Settings > Run in terminal** to try to force your application to run inside a separate terminal, this might be necessary if you try to use **stdin** with e.g. **scanf()** (however it might not work all the time, especially when using the debugger). It might be also because **stdout** full buffering is enabled (i.e. characters are not flushed immediately), you can disable this behavior by e.g. adding **setbuf(stdout, NULL)** in the beginning of your program.
- Use **CTRL+SPACE** to get propositions of **auto code completion**.
- **Pause** during one or two seconds **the mouse above a variable or function** to get information on it.
- **Right-click** on a function or variable and choose **Follow Symbol Under Cursor** to see the corresponding declaration code in the source file.
- When opening an existing source file in Qt Creator, you might be asked to **select the encoding** to be able to edit it. If the file is likely to come from a Windows computer, select the **Windows Codepage 1252** in the list. If it comes from a Linux computer, select **Unicode UTF-8**.
- If the text in a source file is not colored as usual, try closing and reopening the file, and check if it is not asking to select the encoding.
- Depending on the other software installed on the computer (e.g. if Visual Studio 2017 is not installed), you might need to change the **Qt project configuration** to **Release** or **Debug** to run your program successfully.

Tricks/common problems OpenCV

- On Windows, OpenCV might not be configured to be easily used by CMake: you might need to create **OPENCV_DIR** environment variable with **C:\OpenCV4.6.0** value and restart. Also, check in Windows **PATH** for something similar to **C:\OpenCV4.6.0\x86\vc17\bin** and restart if you need to add it.
- Depending on the functions you need, check all the libraries **opencv_XXX.lib** you need to add to the project settings.
- Do not call **cv::Mat::release()/cvReleaseImage()** on an **cv::Mat/IplImage** returned by **cv::VideoCapture::read()/cvQueryFrame()**.
- Be careful to check the type and dimensions of an image returned by **cv::VideoCapture::read()/cvQueryFrame()**, they might be unusual depending on the characteristics of the camera.
- Always use **cv::waitKey()/cvWaitKey()** somewhere after **cv::imshow()/cvShowImage()** to display an **cv::Mat/IplImage** in a window, otherwise the image might not be displayed.
- **cv::waitKey()/cvWaitKey()** might not be able to take into account directly uppercase letters.
- If a camera looks unexpectedly slow, try the suggestions in <https://www.ensta-bretagne.fr/lebars/Share/VideoWebcamOpenCV.zip>.
- Although several samples use the C API, most of the new functionalities of OpenCV are now in its C++ API. Version 4 requires C++11.
- See also https://www.ensta-bretagne.fr/lebars/tutorials/Complements_C-C++.pdf.

Test

- Using a Qt project file (**.pro**): https://www.ensta-bretagne.fr/lebars/Share/ImageOpenCV460_Qt6.4.2.zip for Ubuntu 24.04, https://www.ensta-bretagne.fr/lebars/Share/ImageOpenCV454_Qt6.2.4.zip for Ubuntu 22.04 or https://www.ensta-bretagne.fr/lebars/Share/ImageOpenCV420_Qt5.15.2.zip for Ubuntu 20.04.
- Recent versions of Qt Creator can open **CMake** projects (note that you need to install CMake and restart if not already done...) so it should be possible to use the samples from e.g. https://www.ensta-bretagne.fr/lebars/Share/setup_opencv_Ubuntu.pdf. Launch Qt Creator and open **CMakeLists.txt** as a project, choose **Desktop MinGW 13.2.0 64 bit** configuration (need to be consistent with what was installed and set in the **PATH**) and click **Manage...** button, add **OpenCV_ARCH=x64** and **OpenCV_RUNTIME=mingw** in **CMake Configuration** (those explicit settings might be necessary if OpenCV mixes 64 bit libraries with the 32 bit compiler...). If later CMake makes a warning about **sh** command, try setting **CMake generator** (corresponds to **-G** option of **cmake** command) to **MSYS Makefiles** instead of **MinGW Makefiles**, or ensure any MSYS installation is not in the **PATH**... You might want also to change the **working directory of the application** in the project **Run Settings** so that **image.png** can be found without specifying its absolute path in the C++ code.
- There seem to be a problem with debugging on Windows with MinGW 11.2.0, if needed try to download <https://webperso.ensta.fr/packages/choco/mingw-patch.11.2.0.20230603.nupkg> and from the folder where it is downloaded run **choco upgrade -y -s . mingw-patch --version=11.2.0.20230603**.
- Outside Qt Creator, CMake might not find automatically Qt, see e.g. <https://stackoverflow.com/questions/71086422/cmake-cannot-find-packages-within-qt6-installation> .