# C/C++ programming with Qt 5.11.0 and OpenCV 3.2.0

## Preparation of the computer

- Download **http://download.qt.io/archive/qt/5.11/5.11.0/qt-opensource-windows-x86-5.11.0.exe** and **http://www.ensta-bretagne.fr/lebars/Share/OpenCV3.2.0.zip** , run Qt installer and select **Qt\Qt 5.11.0\MinGW 5.3.0 32 bit** and **Qt\Tools\MinGW 5.3.0 32 bit** options and extract **OpenCV3.2.0.zip** in **C:** (check that the extraction did not create an additional parent folder (we need to get only C:\OpenCV3.2.0\ instead of C:\OpenCV3.2.0\OpenCV3.2.0\), **right-click** and choose **Run as administrator** if needed). For Linux or macOS, additional/different steps might be necessary depending on the specific versions (and the provided .pro might need to be tweaked), see https://www.ensta-bretagne.fr/lebars/Share/setup_opencv_Ubuntu.pdf ; corresponding OpenCV sources : https://github.com/opencv/opencv/archive/3.2.0.zip ; Qt Linux 64 bit : https://download.qt.io/archive/qt/5.11/5.11.0/qt-opensource-linux-x64-5.11.0.run (for Ubuntu you can try **sudo apt install qtcreator qt5-default build-essential** but the version will probably not be the same); Qt macOS : https://download.qt.io/archive/qt/5.11/5.11.0/qt-opensource-mac-x64-5.11.0.dmg .
- In Windows Explorer, right-click on **Computer**, choose **Properties**.
- In the **System** window, click on **Advanced system parameters**. If you do not have administrative rights, on Windows 10 you can press the Windows button, type **path**, and choose **Edit the system environment variables for your account** in the search results to directly access the **Environment variables** window.
- In the **System Properties** windows, click on **Environment variables**.
- In the **Environment variables** window, double-click on the **PATH** variable and add in the end of the **Value** part (without deleting its initial content and add the semi-colons!) **;C:\OpenCV3.2.0\x86\mingw5\bin;**
- Restart. Sometimes, it is necessary to change project configuration to Debug or Release if the project does not run properly in one or the other configuration…

## Tricks/common problems Qt

- Note that when you run your application from Qt, the **working directory of the application** is set by default to the folder that ends with **-build-desktop** (check project **Run Settings** to change that).
- **Create projects only on local disks** (e.g. in C:\TEMP\OPENCV\). Network disks are not fully supported.
- **Delete** the generated files **.pro.user** and the folder that ends with **-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 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…).
- 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, 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 2008 Pro 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

- Depending on the functions you need, check all the libraries **opencv_XXX.lib** you need to add to the project settings.
- Do not call **cvReleaseImage()/cv::Mat::release()** on an **IplImage/cv::Mat** returned by **cvQueryFrame()/cv::VideoCapture::read()**.
- Be careful to check the type and dimensions of an image returned by **cvQueryFrame()/cv::VideoCapture::read()**, they might be unusual depending on the characteristics of the camera.
- Always use **cvWaitKey()/cv::waitKey()** somewhere after **cvShowImage()/cv::imshow()** to display an **IplImage/cv::Mat** in a window, otherwise the image might not be displayed.
- Although several samples use the C API, most of the new functionalities of OpenCV are now in its C++ API. Version 4 is C++11-only.
- See also https://www.ensta-bretagne.fr/lebars/tutorials/Complements_C-C++.pdf .

# Test

- Using a Qt project file (**.pro**): http://www.ensta-bretagne.fr/lebars/Share/ImageOpenCV320_Qt5.11.0.zip .
- 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 . However on Windows, OpenCV might not be configured to be easily used by CMake: you might need to create **OPENCV_DIR** environment variable with **C:\OpenCV3.2.0** value and restart. Then, launch Qt Creator and open **CMakeLists.txt** as a project, choose **Desktop MinGW 5.3.0 32 bit** configuration (to

be consistent with what was set in the **PATH** and what is installed) and click **Manage…** button, add **OpenCV_ARCH=x86** and **OpenCV_RUNTIME=mingw5** 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** to **MSYS Makefile**s 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.