Programmation C/C++ avec Qt 5.5.1 et OpenCV 3.0.0

Préparation de l'ordinateur

- Télécharger **http://download.qt.io/official_releases/qt/5.5/5.5.1/qt-opensource-windows-x86-mingw492-5.5.1.exe** et **http://sourceforge.net/projects/opencvlibrary/files/opencv-win/3.0.0/opencv-3.0.0.exe/download** , lancer l'installation de Qt avec les options par défaut et extraire OpenCV dans **C:\** (penser à lancer les exécutables en faisant un **clic droit** et **Exécuter en tant qu'administrateur**)
- Sur le Bureau (ou Menu Démarrer ou dans l'Explorateur Windows), clic droit sur **Ordinateur**, choisir **Propriétés**
- Dans la fenêtre **Système** ouverte, cliquer sur **Paramètres système avancés**
- Dans la fenêtre **Propriétés système** ouverte, cliquer sur **Variables d'environnement**
- Dans la fenêtre **Variables d'environnement** ouverte, double-cliquer sur la variable nommée **PATH** et rajouter dans la partie **Valeur** à la fin (sans supprimer son contenu initial et bien rajouter les points virgules!)
  ;C:\opencv\build\x86\mingw\bin;C:\opencv\build\x86\vc12\bin;
- Redémarrer. Il faut parfois aussi changer la configuration des projets en Debug ou Release si vos projets ne s'exécutent pas correctement dans l'une ou l'autre de ces configurations (installer Visual Studio 2013 Redistributable (voir https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads) ou même peut-être Visual Studio 2013 Community (voir https://go.microsoft.com/fwlink/?LinkId=532496&type=ISO&clcid=0x409) peut résoudre ce type de problème. Il est cependant probable que seule l'API C d'OpenCV fonctionne (il faudrait recompiler…)…

Tricks/common problems Qt

- Note that when you run your application from Qt, the **current directory of the application** is set by default to the folder that ends with **-build-desktop**.
- **Create projects only on local disks** (e.g. in C:\TEMP\OPENCV\). Network disks are not totally 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 advanced GUI functions) are required, as well as optional data that might be specific to your application (e.g. images to process…).
- 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.
- 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 can 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.

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.
- See also https://www.ensta-bretagne.fr/lebars/tutorials/Complements_C-C++.pdf .

Test

http://www.ensta-bretagne.fr/lebars/Share/VideoWebcamOpenCV300_Qt5.5.1.zip