C/C++ programming with Visual Studio 2017 and OpenCV 4.2.0

Preparation of the computer

- Download **Visual Studio 2017 Community** (be sure to install the necessary workloads and components, e.g. with this command: vs_community.exe --passive --norestart --wait --addProductLang en-US --includeRecommended --add Microsoft.VisualStudio.Component.CoreEditor --add Microsoft.VisualStudio.Workload.NativeDesktop --add Microsoft.VisualStudio.Workload.ManagedDesktop --add Microsoft.VisualStudio.ComponentGroup.NativeDesktop.WinXP --add Microsoft.VisualStudio.Component.WinXP --add Microsoft.VisualStudio.Component.VC.Tools.x86.x64 --add Microsoft.VisualStudio.Workload.NativeCrossPlat --add Microsoft.VisualStudio.Component.VC.ATLMFC --add Microsoft.VisualStudio.Component.VC.CLI.Support --add Microsoft.VisualStudio.Component.VC.Modules.x86.x64 --add Microsoft.VisualStudio.Workload.Python --remove Microsoft.Component.CookiecutterTools --remove Microsoft.Component.PythonTools.Web --remove Component.CPython3.x64 --add Microsoft.VisualStudio.Component.Graphics --add Component.GitHub.VisualStudio --add Microsoft.VisualStudio.Component.JavaScript.Diagnostics --add Microsoft.VisualStudio.Component.JavaScript.TypeScript --add Microsoft.VisualStudio.Component.TestTools.Core --add Component.Linux.CMake, note also that the first release of Visual Studio 2017 does not have Python support, check also the required prerequisites) and **http://www.ensta-bretagne.fr/lebars/Share/OpenCV4.2.0.zip** (contains OpenCV with extra modules built for Visual Studio 2015, 2017, 2019, MinGW Qt 5.12.6 x86, MinGW 8 x64), install Visual Studio with **Desktop development with C++ Workload** and extract OpenCV4.2.0.zip in **C:** (check that the extraction did not create an additional parent folder (we need to get only C:\OpenCV4.2.0\ instead of C:\OpenCV4.2.0\OpenCV4.2.0\), run as administrator if needed).
- 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:\OpenCV4.2.0\x86\vc15\bin;**
- Restart.
- If needed, see http://www.ensta-bretagne.fr/lebars/tutorials/screenshots_vs2015_cv249_win10.pdf and http://www.ensta-bretagne.fr/lebars/tutorials/Complements_C-C++.pdf for more information.

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 Visual Studio project file (**.vcxproj**): http://www.ensta-bretagne.fr/lebars/Share/ImageOpenCV420_vs2017.zip .
- Recent versions of Visual Studio can open **CMake** projects 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:\OpenCV4.2.0** value and restart. Then, launch Visual Studio and go to **File\Open\CMake…** menu and open **CMakeLists.txt**, add an **x86 Release** configuration (to be consistent with what was set in the **PATH**). You might want also to right-click on **CMakeLists.txt** in the **Solution Explorer**, choose **Debug and Launch Settings** and add **"currentDir": "${workspaceRoot}"** in the **launch.vs.json** file, so that **image.png** can be found without specifying its absolute path in the C++ code.