C/C++ programming with Visual Studio and OpenCV versions from package manager

Preparation of the computer

- Install **Chocolatey** package manager: **https://chocolatey.org/install** and then install **CMake** with **choco install -y cmake.install --installargs 'ADD_CMAKE_TO_PATH=System'**.
- Install **Visual Studio 2022 Community** (be sure to install the necessary workloads and components, e.g. with this command: **choco install -y visualstudio2022community --package-parameters "--addProductLang en-US --includeRecommended --add Microsoft.VisualStudio.Component.CoreEditor --add Microsoft.VisualStudio.Workload.NativeDesktop --add Microsoft.VisualStudio.Workload.ManagedDesktop --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 --remove Component.Microsoft.VisualStudio.LiveShare --remove Microsoft.VisualStudio.Component.VC.TestAdapterForGoogleTest --remove Microsoft.VisualStudio.Component.VC.TestAdapterForBoostTest --remove Microsoft.VisualStudio.Component.EntityFramework --add Microsoft.VisualStudio.Workload.Python --remove Microsoft.Component.CookiecutterTools --remove Microsoft.Component.PythonTools.Web --remove Component.CPython3.x64 --remove Microsoft.Component.PythonTools.Minicondax64 --add Microsoft.VisualStudio.Component.Graphics --add Microsoft.VisualStudio.Component.JavaScript.Diagnostics --add Microsoft.VisualStudio.Component.JavaScript.TypeScript --add Component.Linux.CMake"**, check also the required prerequisites).
- Install **OpenCV** package with **choco install -y libopencv-dev --version=4.5.4.20240203 --params "'/url1:https://github.com/lebarsfa/Packages/releases/download/libopencv-dev.4.5.4.20220805/libopencv-dev.4.5.4_x86_vc17_staticlib_Debug.exe /url2:https://github.com/lebarsfa/Packages/releases/download/libopencv-dev.4.5.4.20220805/libopencv-dev.4.5.4_x86_vc17_lib_Debug.exe /url3:https://github.com/lebarsfa/Packages/releases/download/libopencv-dev.4.5.4.20220805/libopencv-dev.4.5.4_x86_vc17_staticlib_Release.exe /url4:https://github.com/lebarsfa/Packages/releases/download/libopencv-dev.4.5.4.20220805/libopencv-dev.4.5.4_x86_vc17_lib_Release.exe'"**. You might also want to prevent further updates with **choco pin add -n libopencv-dev**.
- 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

- On Windows, OpenCV might not be configured to be easily used by CMake: you might need to create **OPENCV_DIR** environment variable with **C:\OpenCV4.5.4** value and restart. Also, check in Windows **PATH** for something similar to **C:\OpenCV4.5.4\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.
- 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 Visual Studio project file (**.vcxproj**): http://www.ensta-bretagne.fr/lebars/Share/ImageOpenCV454_vs2022.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. 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 installed and 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. See also https://www.ensta-bretagne.fr/lebars/tutorials/vs_cmake.txt.