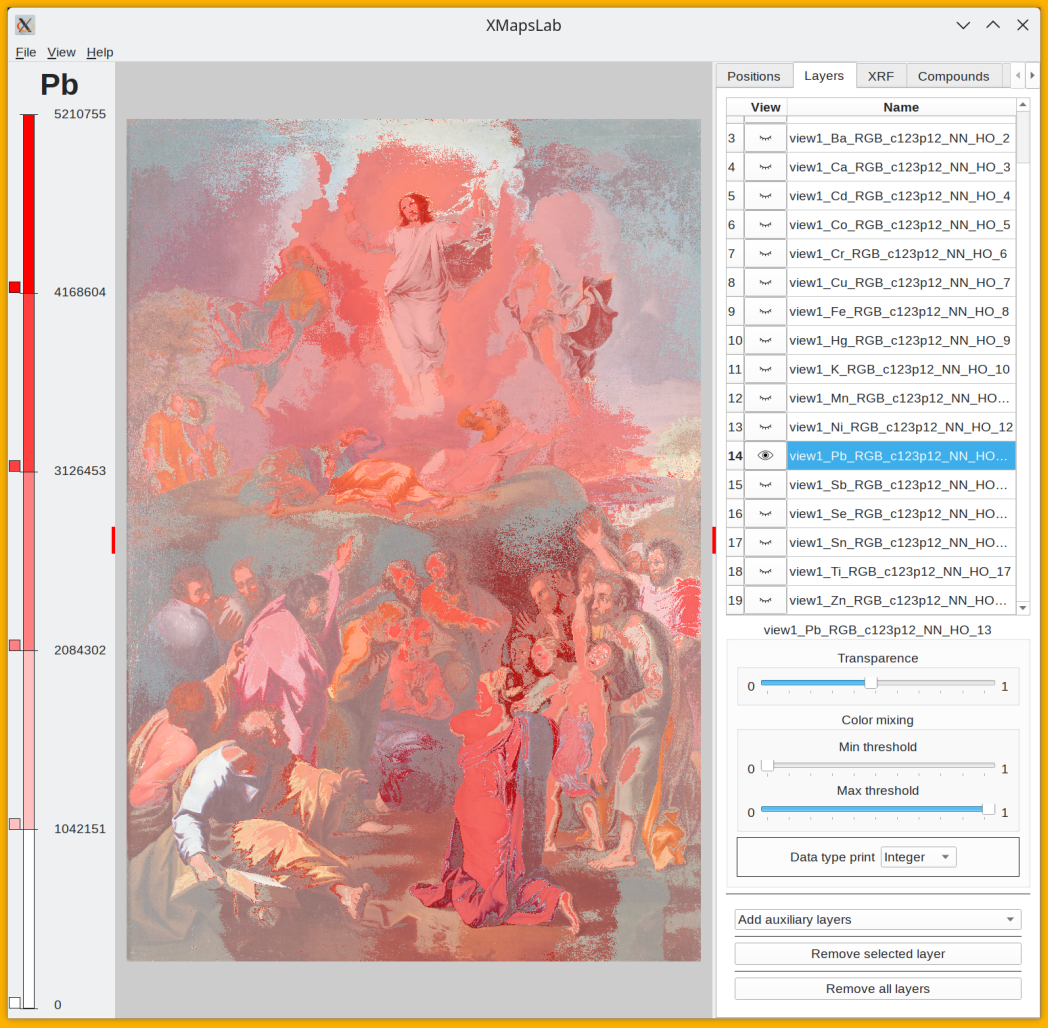


Compiling XMapsLab

Domingo Martín Perandrés
dmartin@ugr.es
https://calipso.ugr.es/xmapslab.org

2023

Contents

List of Figures

1

Compilation process in Linux

This chapter shows the steps to accomplish for compiling **XMapsLab** in Linux.

The compilation of **XMapsLab** implies that some programs, libraries and headers are installed and running correctly, particularly the OpenCV and GLEW libraries and , of course, the Qt libraries and programs. The explanation assumes an Ubuntu-based distribution but it will be similar in others.

We can have a fresh or not fresh installation of the software. For the explanation we will assume that a fresh installation is done.

The first step is to get the OS installed. In the case of Ubunutu and derivations, that implies to download an .iso file that will be used to do all the process. One of the important things to do is the installation of the graphics drivers for the GPU. This implies that OpenGL and other related libraries will be installed.

Before installing the other needed libraries and software it is important to take care about the type of installations: system-wide or in private folders. The easy way is to install everything system-wide. In such case, the searching paths for the libraries are predefined and it is not necessary to include them in the project files. In case that some of the libraries, OpenCV and GLEW, are installed in local folders, the project must include the paths to them (headers and libraries).

After having the OS running, we must add the programs to compile and link programs, so it is necessary to install g++, make, etc.

The command to get the compiler, linker, etc. is:
sudo apt-get install build-essential

While the drivers and libraries of OpenGL should be installed from the begining, the headers could be missed. Particularly, we need the gl.h file. All the OpenGL related headers must be in the folder /usr/include/GL.

You can try this:

1. sudo apt-file search gl.h (it will show the packets that have the required file)
2. sudo apt-get install libmesa-dev

1.1 GLEW

GLEW is a library that allows to identify and load OpenGL extensions.

The installation can be easily done by following this sequence of commands:

1. Dowload the last version from https://glew.sourceforge.net/
2. Uncompress the file: gunzip glew-x.x.x.zip (In case you have not installed unzip you will do it with this command: sudo apt install unzip)
3. Change to the GLEW folder: cd glew-x.x.x
4. Modify the Makefile (if necessary, GLEW_PREFIX and GLEW_DEST for changing the folders where the files will be installed)
5. Compile: make
6. Install: make install

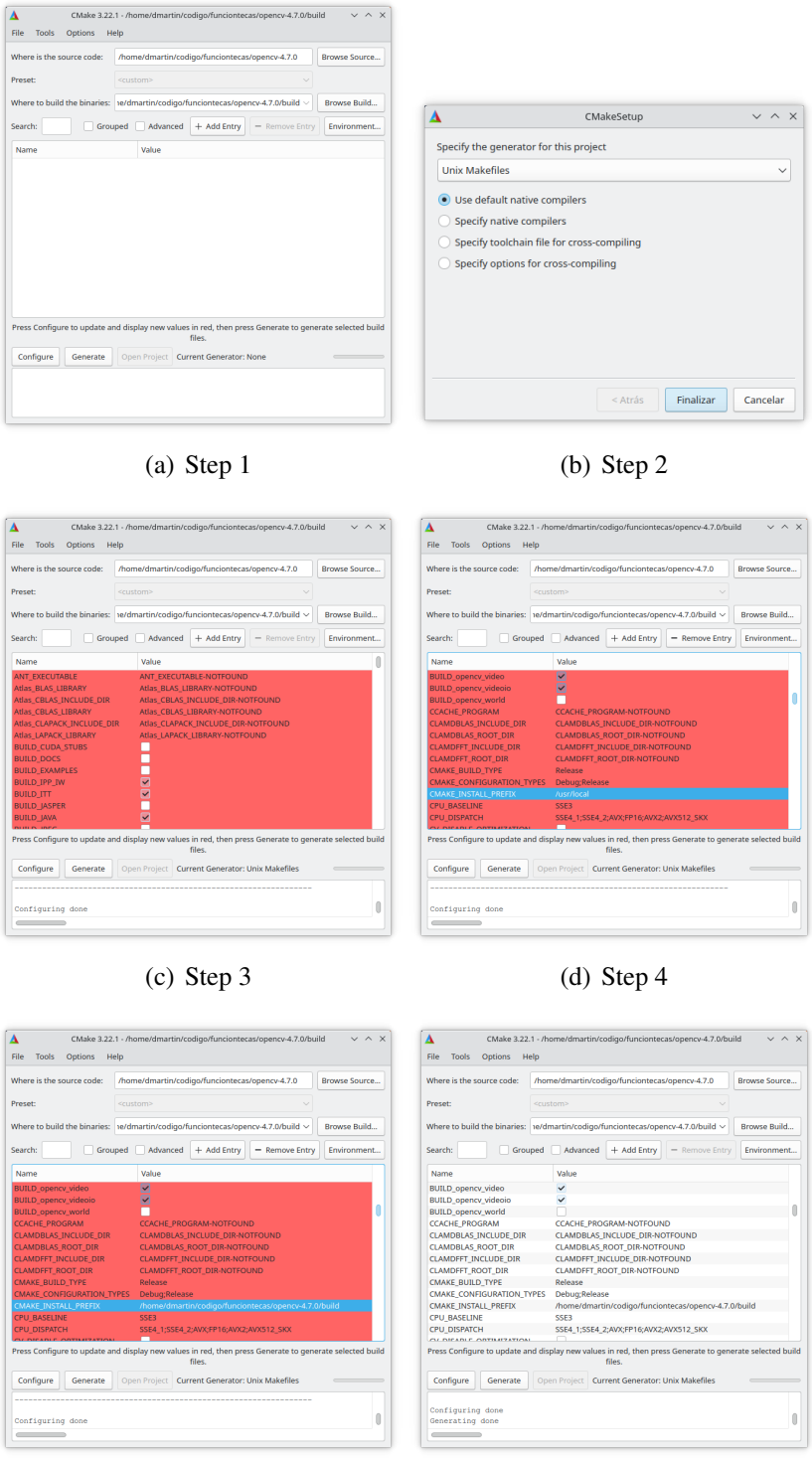
The headers of GLEW can be copied to /usr/include/GL if we want that they are visible for all users.

1.2 OpenCV

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly for real-time computer vision. We use it mainly for doing some operations on the images.

For compiling the OpenCV library it is necessary to install cmake and cmake-gui. This is done with:
sudo apt install cmake
sudo apt install cmake-gui

The next step is to download the OpenCV code from https://opencv.org/ (Figure 1.2(e)). We must click on the Library option in the top menu and select Release (Figure 1.2(e)). A file called opencv-x.x.x.zip will be donwloaded. We must move this file to a folder where to do the compilation. Then the file is uncompressed with: gunzip opencv-x.x.x.zip



(a) Step 1 (b) Step 2

(c) Step 3 (d) Step 4

(e) Step 5 (f) Step 6

Figure 1.1: Steps for using cmake-gui.

We will change to the folder opencv-x.x.x with cd opencv-x.x.x and we will create a folder called, for example, build. All the files of the compilation will be put in this folder.

Now we have to create the Makefile files using cmake-gui. Run it with the command cmake-gui. The first thing is to indicate where is the CMakeLists.txt file: we must select the opencv-x.x.x folder. Then we must indicate where the build files will be put: we must select the build folder that we have previously created (Figure 1.1(a)).

Then we must click on the Configure to create a configuration. The program will ask for the type of process we want. We will select Unix Makefiles and will check to use the default native compilers (Figure 1.1(b)). The found parameters and their values will be shown (Figure 1.1(c)). One important parameter is the folder where the library will be installed (do not confuse with the folder where it will be compiled). Once possibility is to leave as system-wide (Figure 1.1(d)) or to place it in other folder, as is shown in Figure 1.1(e). Once all the needed changes are achieved, the Makefile files are created by clicking on Generate button (Figure 1.1(f)).

Finally we will change to the folder build with cd build and we will compile with make[1] command and then install in the selected folder with make install command.

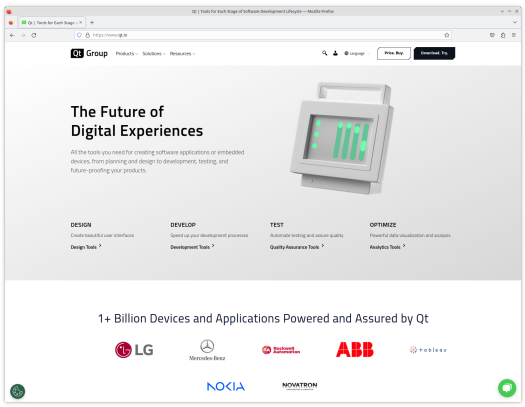
1.3 Qt

Qt is a multi-platform Graphical User Interface (GUI) framework written in C++. The main advantage is that a single code can be used to produce applications for multiple Operating Systems and platforms. We use the Qt library and also QtCreator, the development program.

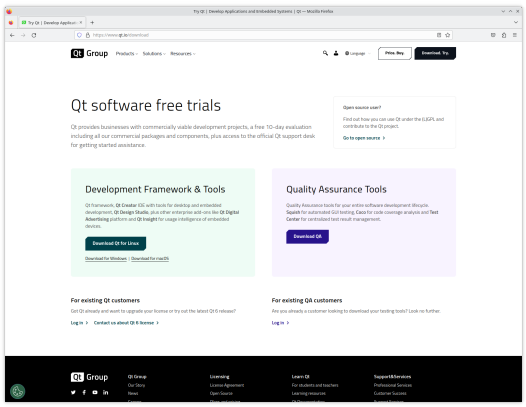
The first step for loading Qt is to go to the web page https://www.qt.io/ and click on the Download.Try button (top right) (Figure 1.2(a)). Then you have to click on a small link Go to open source that is on a box on the right side (Figure 1.2(b)). You will see the image shown in Figure 1.2(c), but you have to go down until see the image shown in Figure 1.2(d). Then, click on the Download the Qt online installer green button. You will have the option of seleccting the appropriate version (Figure 1.2(e)). We will select the Linux version. A file called qt-unified-linux-x64-x.x.x-online.run will be loaded. This is the script that must be run to install Qt.

It is important to note that the first time it is necessary to register. This can be done by clicking in the link Register (Figure 1.3(a)). In the second step the bottom boxes should be marked (Figure 1.3(b)). The third step is for requesting the help of the user to improve the programs (Figure 1.3(c)). In the fourth step we must select the folder where Qt will be installed and we must select how the installation will be. We will select a custom installation (Figure 1.3(d)). We will select the LTS version by clicking on the box and then on the Filter button. We will select the Qt 6.x.x (6.2.4 in the moment of writing this) as shown in Figure 1.3(e) and Figure 1.3(f). The licence must be accepted (Figure 1.3(g)). In the last

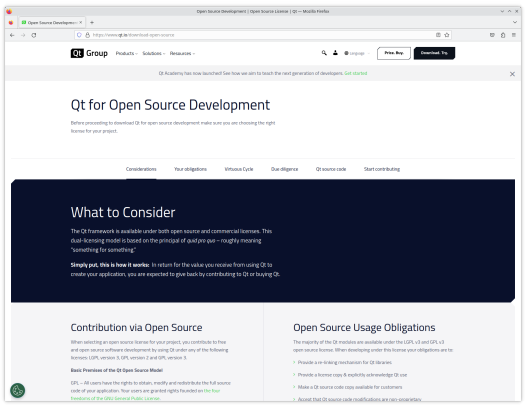
[1] If we want a faster compilation the -jN option will be added with n being the number of threads to use



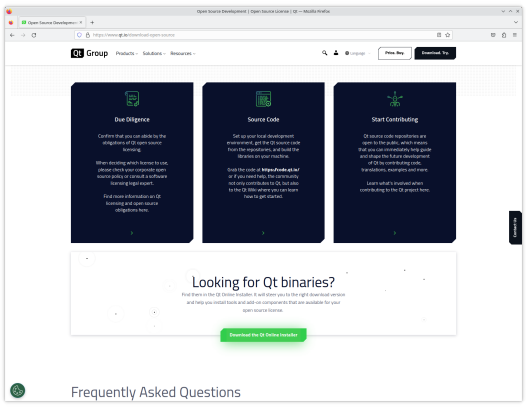
(a) Step 1



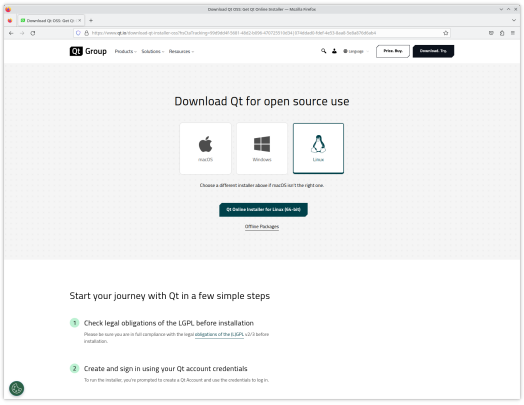
(b) Step 2



(c) Step 3



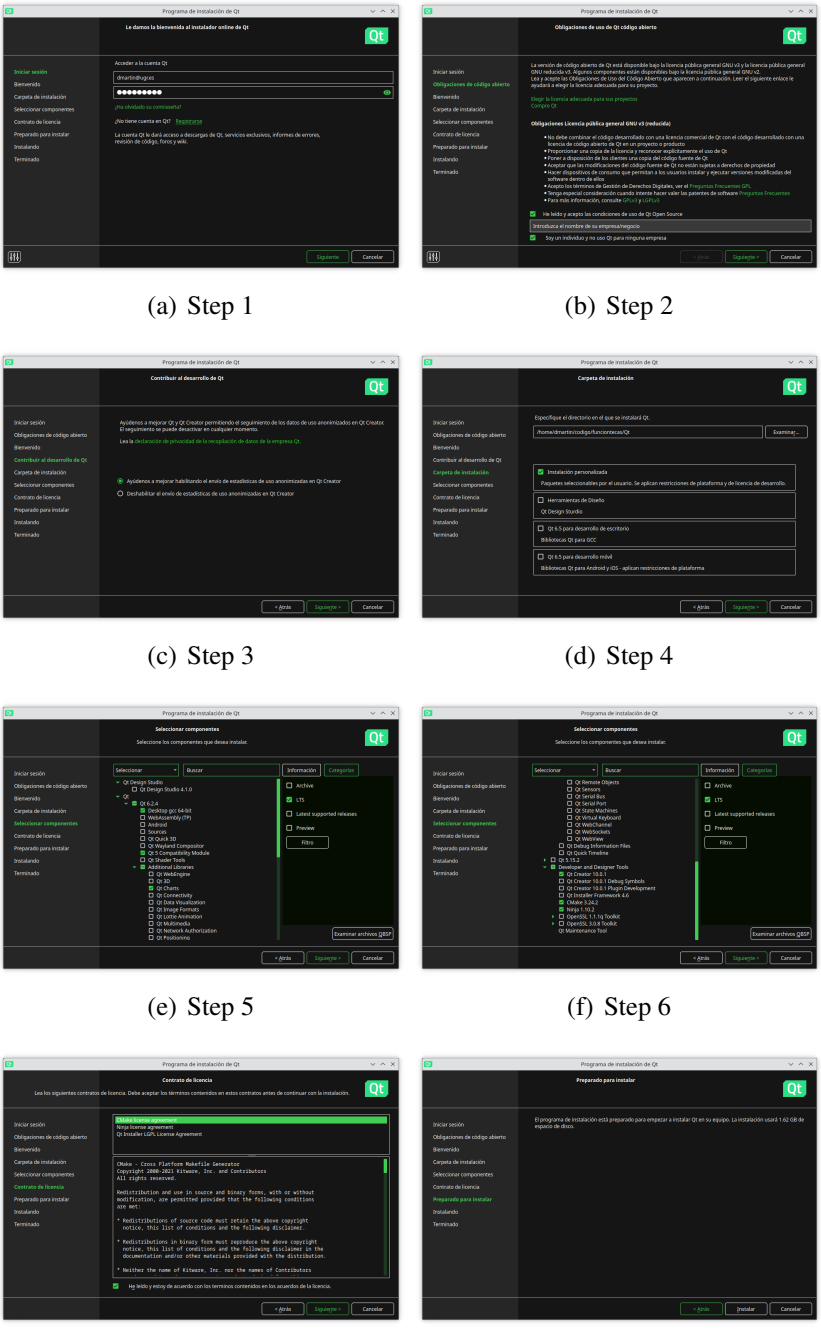
(d) Step 4



(e) Step 5

Figure 1.2: Steps for loading Qt.

image (Figure 1.3(h)) we will select the Install button. At the end of the process we could run QtCreator program.



(a) Step 1 (b) Step 2

(c) Step 3 (d) Step 4

(e) Step 5 (f) Step 6

(g) Step 7 (h) Step 8

Figure 1.3: Steps for installing Qt.

1.4 Compiling XMapsLab

Once we have Qt installed we can compile the **XMapsLab** program. These are the step:

1. Dowload the source files of **XMapsLab**
2. Uncompress it
3. Run QtCreator
4. Load the project: Select File → Open File or Project (search for the xmapslab.pro file, select it and click on the open button (or double click on the file)).
5. Configure the project. Select the kit. Check the box Desktop Qt 6.2.4 GCC 64 bit
6. Click on Configure project
7. Click of the icon of the wrench (Projects). Uncheck the box Shadow build (you can read more about shadow build and take your own decisions; we do this to avoid to copy folders to the shadow build folder)
8. Click of the the Edit icon. The Projects tab (top-left) will show the name of the project. Click on the small triangle to show the folder and file structure.
9. Double click on the name stippleshop.pro name. This will open the file in the editor
10. Modify it with your preferences. The configuration of the xmapslab.pro must be changed to match the own installation, being particularly careful with the path (see below).
11. Save it.
12. Build the project by pressing CTRL+b or by clicking on the single green triangle (like a play button) to build and run.
13. Use **XMapsLab**

```
DEFINES+=LINUX
#DEFINES+=WINDOWS


# PATHS
LINUX_PATH=/XXX/YYY/ZZZ
WINDOWS_PATH=C:\UUU\

MAIN_PATH=src/main
COMMON_CLASSES_PATH=../common/common_classes
RBF_PATH=src/rbf
SHADERS_PATH=../common

linux {
TARGET= xmapslab

INCLUDEPATH += $${LINUX_PATH}/opencv-4.6.0/include/opencv4
INCLUDEPATH += $${LINUX_PATH}/opencv-4.6.0/include/opencv4/opencv2

INCLUDEPATH += $${LINUX_PATH}/glew-2.1.0/include

INCLUDEPATH += $${MAIN_PATH}
```

```
INCLUDEPATH +=$${FILE_IO_PATH}
INCLUDEPATH +=$${COMMON_CLASSES_PATH}
INCLUDEPATH +=$${RBF_PATH}
INCLUDEPATH +=$${SHADERS_PATH}

INCLUDEPATH += $${LINUX_PATH}/Eigen

LIBS += \
    -L$${LINUX_PATH}/glew-2.1.0/lib -lGLEW \
  -L$${LINUX_PATH}/opencv-4.6.0/lib -lopencv_core -lopencv_highgui \
  -lopencv_imgproc -lopencv_imgcodecs \
  -lGL
}
```

This same process can be used to compile **Positions** and **Tools**, adjusting the ".pro" files accordingly.

Although the description that has been made is for compilation on a Linux distribution like Ubuntu, the .pro files are ready to compile on Windows just by changing the target commenting DEFINES+=LINUX and uncommenting DEFINES+=WINDOWS.