QT and OpenCV

Phil Culverhouse

QT is a cross-platform development environment. Fig.1 shows how to open it from the start menu. If you want to download it for your laptop see Annex A for more information.

Figure 1: QT menu (Community)

Once QT has opened there are two default projects. Select OpenCV_Samples and open that project (Fig.2)

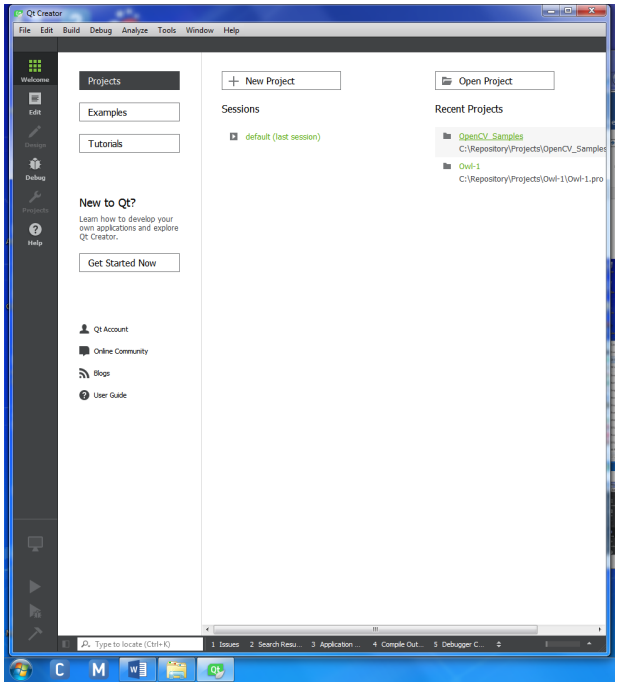

Figure 2: QT base menu

You can compile (build) and run the sample code (the source is called contoures2.cpp). See Fig. 3 to see the code and the windows that it opens when run.

Note the slider bar on the right hand window allows user input to control the run time code. Check out where this is happening inside the source code.

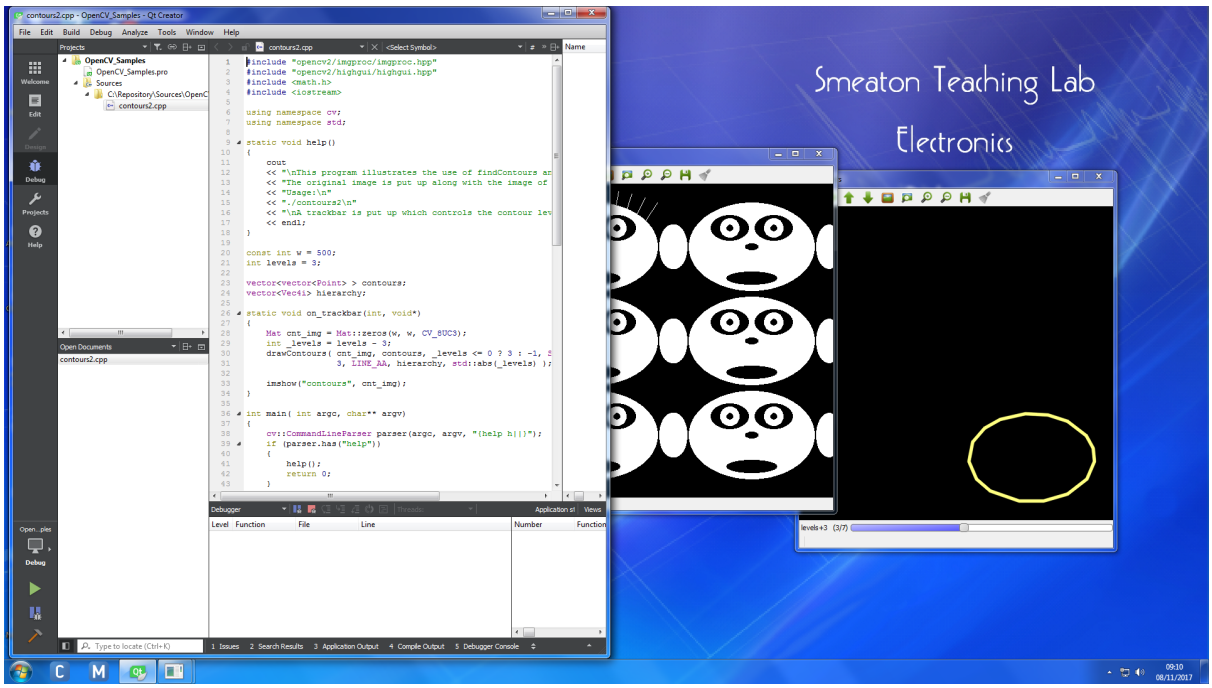

Figure 3: OpenCV_samples code running, showing source

If you want to create a new project, the way I suggest is to have a repository as we have set on each machine. Project details are kept separate from sources, so that code can easily be re-used across projects. You can make your own repository on another drive if you wish.

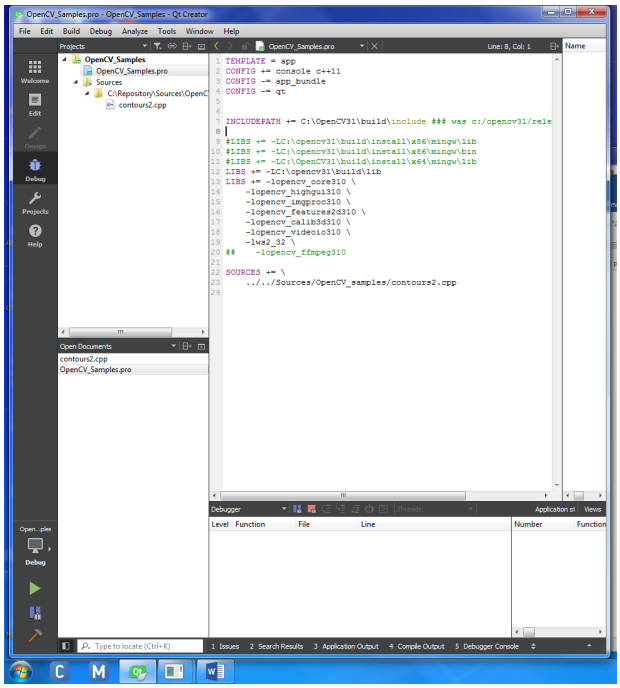

Figure 4: the QT .pro file

New projects can be created using the QT menu. Select Non-QT C++ project, select your repository/Project folder and give the project a name. You can then add sources to the project late4r. By default, QT puts a main.cpp in the project Named folder. This is probably best to delete this and add your own that is in a Project-related source folder. See Fig. 5 for our example of a repository contents.

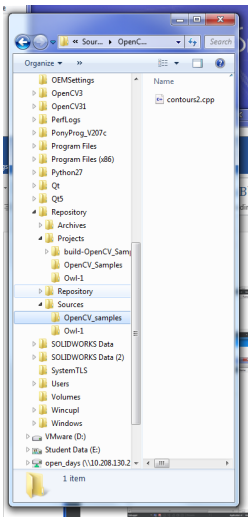

Figure 5: Repository

The .PRO file (held in the project folder) contains definitions that the compiler needs to build the project. A new project will not have the correct settings for OpenCV, so copy these from the example (or from Fig.6 below). Notice they are specifying the locations of libraries and include files. The file paths are different for linux, PC, Mac.

```
TEMPLATE = app
CONFIG += console c++11
CONFIG -= app_bundle
CONFIG -= qt

INCLUDEPATH += C:\OpenCV31\build\include ### was c:/opencv31/release/install/include

#LIBS += -LC:\opencv31\build\install\x86\mingw\lib
#LIBS += -LC:\opencv31\build\install\x86\mingw\bin
#LIBS += -LC:\OpenCV31\build\install\x64\mingw\lib

### Change to suit version of openCV
LIBS += -LC:\opencv31\build\lib
LIBS += -lopencv_core310 \
    -lopencv_highgui310 \
    -lopencv_imgproc310 \
    -lopencv_features2d310 \
    -lopencv_calib3d310 \
    -lopencv_videoio310 \
    -lopencv_imgcodecs310 \
    -lws2_32 \
##    -lopencv_ffmpeg310
```

Figure 6: QT .pro file default contents

Debugging

Build the project (I recommend using Build>ReBuild project) to ensure all changes to the .PRO file and sources are taken into the latest build. Place a break point at line 84 in the Contours2.cpp source. Run the code in Debug mode, and expand the debug variables window (see Fig. 7 RHS, example shows Contours0 expanded, showing each contour in the vector list, and also shows one Point in the first contour (showing that it is an X,Y pair called a Point (cv::Point). See Fig.8 for the run in debug mode icon (green button with bug). See https://docs.opencv.org/trunk/d0/d2a/contours2_8cpp-example.html

Figure 8: the Debug /run icons

Lookup Points in the online help for OpenCV. Also note that sample code (incl contours2.cpp) are in the openCV sources samples folder (check on your machine where OpenCV is installed. Hint look at the .PRO file for the example project.

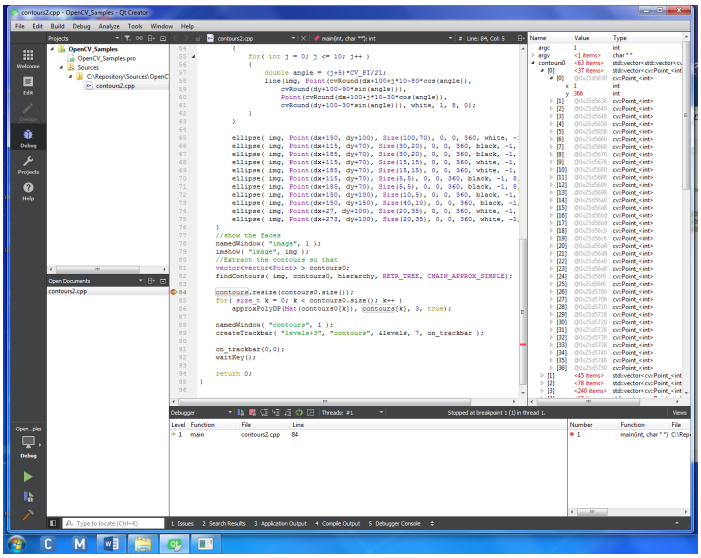

Figure 7: Debugging with QT

Note that you can single step through the code also. But be careful with system call back routines, when trying to single step through code. Call-backs are used to link graphical in/out with user code. See the graphics are created by OpenCV and pushed into a live list of screen drawing things. Your code takes note of the list reference and links it to your code. They may not work properly. So, these are all the screen display functions such as createTrackba ();

Example:

createTrackbar("levels+3", "contours", &levels, 7, on_trackbar);

makes a graphic tracker bar, and associates it with the function on_trackbar().

When the user moves the trackbar with the mouse, the system decodes this as being an interaction with on_trackbar(), calls it and gives it the position of the trackbar via the variable levels.

Note that to display images (or video frames in sequence from a live source) using cv::ImShow() you need to give the display time to get the image from the camera buffers, format it for the screen and display it during the next frame refresh of the screen. This takes about 30ms. So always associate displaying with an OpenCV wait, which is easiest using the cv::waitKey(30); function call.

In general take a look at the extensive on-line documentation. Start here for example https://docs.opencv.org/master/d9/df8/tutorial_root.html

Annex 1:

Installing QT/OpenCV on your laptop.

There are several issues to consider.

1) Choice of development language (C++, Python are two popular ones) I prefer C++ for speed.
2) Choice of development environment (QT, Eclipse, Visual Studio are popular). I choose QT as its cross platform and is quicker to learn than Eclipse.
3) Choice of compiler
    a. If Visual Studio, default to VS compiler for selected VS SDK
    b. If QT, and wanting to use OpenCV without VS compiler select Mingw32 or 64 as the posix compliant compiler toolset with gcc.

So, go to QT download, https://www1.qt.io/download-open-source/?hsCtaTracking=f977210e-de67-475f-a32b-65cec207fd03%7Cd62710cd-e1db-46aa-8d4d-2f1c1ffdacea#section-2

This gives all the options.

Select the one for your system. EXCEPT that for windows with mingw32/64 choose this or the 64 bit version. This will load the mingw compiler/debugger kit too (gcc & gdp).

- Qt 5.6.3 for Windows 32-bit (MinGW 4.9.2, 1.0 GB) (info)
- Note that the link here will only take you to 5.6.3 dev kit stuff. If you are doing this later, check the live webpage for the latest.
- Run this installer AND ensure that you select both QT options (except you might not need the graphics libraries, which are very extensive), and the mingw compiler option. Neither are in the defaults. This should amount top about 4.4GB.
- Then get the OpenCV library and build. See https://opencv.org/releases.html
- Choose Sources if you are using MAC, LINUX or MINGW (for windows), and follow the build instructions.
- In the .PRO file you will need to adjust the library and include folder locations.
- If you have selected 32 bit opencv and 64bit QT/mingw (or visa-versa) then you will get a runtime crash. If you have selected VS pre-build opencv binary libraries and try to link them with QT for mingw compiled code you will also get a runtime crash.
- Good Luck.