QGAR Environment

Philippe Dosch

1 General

Since 12 years, the QGAR[1] project-team has devoted much effort to the construction of a software environment, to be able to reuse whole or part of software implemented during previous work, as well as collected experience. It includes the works of some researchers, PhD students and engineers produced during some PhD thesis and industrial contracts. The system has been used within the context of several cooperation projects, including European projects.

QGAR is a registered trademark of INRIA, and the QGAR environment is registered as free software by the French agency for software protection (APP), according to the terms of two licenses:

- the GNU Lesser General Public License (LGPL) for the installation/compilation tools, the core library QgarLib, the applications globally so-called QgarApps,
- the Q Public License (QPL) for the the graphical user interface QgarGui.

It may be freely downloaded from its web site (`http://www.qgar.org`).

2 Content

The QGAR environment includes three main parts (see figure1):

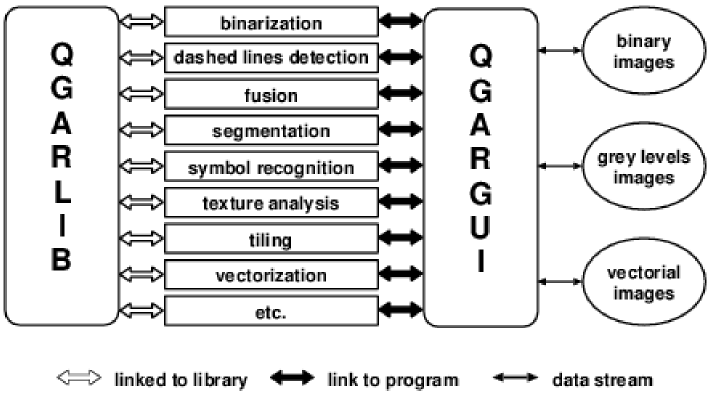

Figure 1: Architecture of the QGAR environment.

[1] *Querying Graphics through Analysis and Recognition.*

- *QgarLib*, a library of C++ classes implementing basic graphics analysis and recognition methods,
- *QgarApps*, an applicative layer, including high-level applications (binarization, edge detection, text-graphics separation, thick-thin separation, vectorization, etc.),
- *QgarGUI*, a graphical interface to design and run applications, providing data manipulation and display capabilities.

2.1 QgarLib

QgarLib is a library including around 150 C++ classes according to different thematics:

- *Image processing*: binarizations (Trier, Niblack, Hysteresis), mathematical morphology, distance transformations, skeletonization, convolutions, Gradients and Laplacians, edge detection (Canny, Deriche)...
- *Graphical processing*: polygonal approximations, Freeman chains, connected components, vectorization... polylines...
- *Data structures*: images, graphs, trees, histograms, masks...
- *Tools*: files input/output, object serialization, classification...

2.2 QgarApps

The QgarApps are stand-alone applications built with the QgarLib library. These applications can be interactively launched, either from a shell or from the graphical user interface. Around 10 applications are available in the QGAR environment: binarizations, text-graphic separations, thin-thick separations, text extraction, vectorizations, image degradation, symbol recognition...

Each application is described thanks to a XML file allowing to drive it from the GUI.

2.3 QgarGUI

Finally, the QgarGUI interface (see figure 2) allows to drive and to tune applications, to display results, to correct and to fix them.

3 Implementation

The whole system is written in C++ and includes about 170,000 lines of code, including unit test procedures. A particular attention has been paid to the support of "standard" formats (PBM+, DXF, SVG), high-quality documentation, configuration facilities (using CMake), and support of Unix/Linux and Windows operating systems. The GUI is built using the Qt toolbox.

Application management is plugin-based. Each executable binary file is paired with a XML description file which is parsed when the user interface is launched: the corresponding application is then dynamically integrated into the menus of the interface, and dialog boxes to access the documentation and run the application are dynamically generated. In this way,

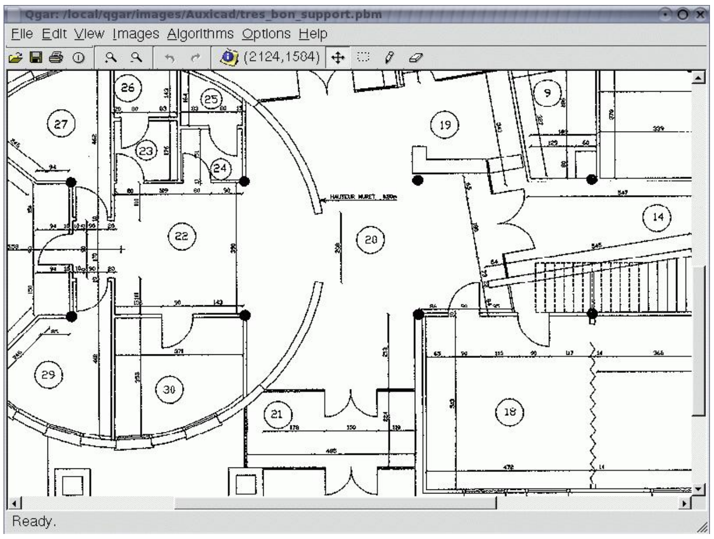
Qgar: /local/qgar/images/Auxicad/tres_bon_support.pbm
File Edit View Images Algorithms Options Help
(2124,1504)
Ready.

Figure 2: Screenshot of QgarGUI.

any application may be easily coupled with a remote system based on a similar approach. Conversely, as the integration (or removal) of an application does not imply any modification of the user interface, the installation of remote applications, provided by partners for testing for example, is easy. This is particularly useful when comparing different methods performing the same task, in the context of performance evaluation, a topic which is part of our current research work, as previously mentioned.

The whole package, including all the components, is proposed under several formats (gzipped tar archive, DEB package, Windows installer) in order to provide different easy ways to install Qgar, on most of existing platforms.

4 Example

An application typically works in several steps, as shown in the example below, implementing the Canny edge detection.

```
// STL
#include <list>

// QGAR
#include <qgarlib/CannyGradientImage.H>
#include <qgarlib/GenImage.H>
#include <qgarlib/GradientLocalMaxImage.H>
#include <qgarlib/HysteresisBinaryImage.H>
#include <qgarlib/PbmFile.H>
#include <qgarlib/PgmFile.H>
#include <qgarlib/QgarApp.H>

using namespace qgar;
using namespace std;

int main(int argc, char* argv[])
{
  QgarApp app;

  // PARAMETERS DESCRIPTION
  // ======================

  // Input file
  app.addParameter("-in",
                   QgarArgs::REQPARAM,
                   QgarArgs::FILEIN,
                   "source image:");

  // Output file
  app.addParameter("-out",
                   QgarArgs::REQPARAM,
                   QgarArgs::FILEOUTD,
                   "result image:",
                   ".pbm");

  // And so on for -low and -high parameters...

  app.setDescription("Edges detection", QgarArgs::PGM);

  // ANALYZE THE COMMAND LINE
  // ========================

  app.analyzeLine(argc, argv);
  if (app.isError()) { return app._CODE_ERROR; }
  if (app.isExit()) { return app._CODE_GUI; }

  // GET SOURCE IMAGE
  // ================

  PgmFile sourceFile((char*) app.getStringOption("-in"));
  GreyLevelImage sourceImg = sourceFile.read();

  // COMPUTE GRADIENT
  // ================

  CannyGradientImage gradImg(sourceImg, 1.2);

  // COMPUTE LOCAL MAXIMA
  // ====================

  GradientLocalMaxImage maxImg(gradImg);

  // HYSTERESIS THRESHOLDING
  // =======================
  HysteresisBinaryImage edgesImg(maxImg,
                               atoi(app.getStringOption("-low")),
                               atoi(app.getStringOption("-high")));

  // SAVE RESULT
  // ===========

  PbmFile resultFile((char*) app.getStringOption("-out"));
  resultFile.write(edgesImg);

  // NORMAL TERMINATION
  // ==================

  return app._CODE_END;
}
```