# Performance et Génericité dans QGAR

Vitor Vasconcelos Araújo Silva


20 février 2008

INRIA LORRAINE

# Le problème

- Analyse de documents : noumbreuses techniques complexes.
- Óperations courants
  - Traitement d'image classique
  - Morphologie mathematique
  - Détection de contours
  - E/S fichiers, (dé)sérialisation
  - Binarisation, etc...
- Chacun code 'ses' algorithmes. Aucune documentation, aucun standard.
- Personne n'utilise le code disponible... Le cycle recommence.

# Encore...

- Évolution de la recherche : nouveaux algorithmes et téchniques doivent être ajoutés au “lot” disponible.
- Pas d’interface avec les outils développés.
- Sauvegarde des résultats.
- Outils déjà dévéloppés!
  - Très géneriques mais difficiles à utiliser.
  - Très spécifiques, pratiques et robustes, plus faciles à utiliser.
  - Mais pour documents à forte teneur graphique, disponibles en petit nombre.

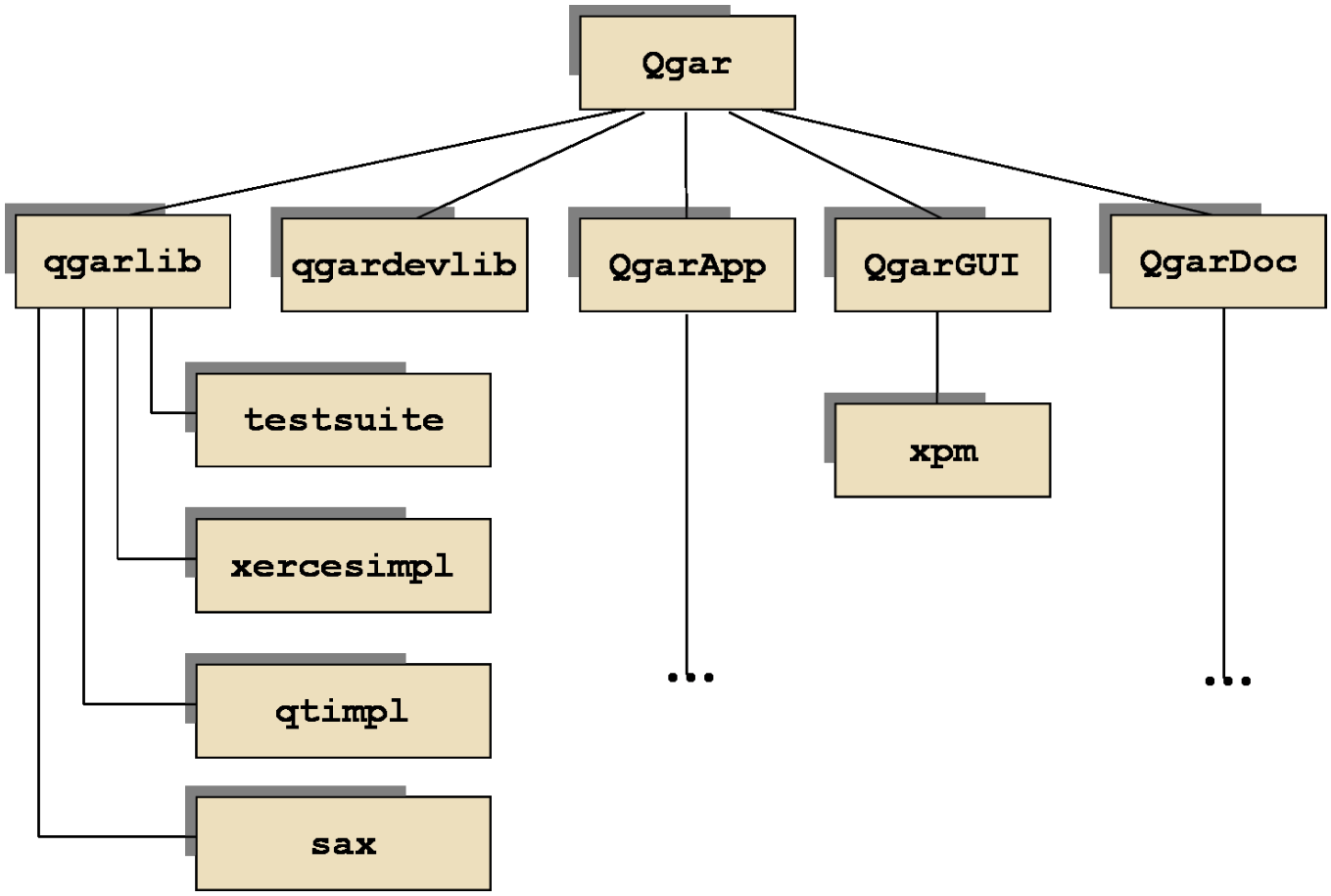
# La solution

- Plateforme développée en focntion des besoins décrits.
- Caractéristiques souhaités:
  - Facilité d'utilisation
  - Interfaçable avec autres systémes
  - Pouvant être continuellement enrichi de nouvelles méthodes
  - Avec une interface utilisateur.
- Standards de formats, outils de développement, développement semi-profissionel. $\Rightarrow$ standard QGAR de développement a éte defini (et suivi!)

# QGAR Software

- Composants de la plate-forme QGAR
  - QgarLib : bibliothèque de base qui offre les fonctions d'analyse graphique et methodes de reconaissance des formes.
  - QgarDevLib : bibliothèque de base dont les modules ne sont pas encore "validés" conforme les standards QGAR.
  - QgarApps : applications autonomes de reconaissance graphique.
  - QgarGUI : interface graphique (codé avec Qt) pour la manipulation et l'affichage des résultats du traitement graphique.

# QGAR Software



# QGAR description

- Codé entierement en C++/Qt.
- 100 000 lignes de code.
- Logiciel libre LGPL/QPL.
- Site web: **www.qgar.org**.
- Code deposé à l'Agence pour la Portection des Programmes
- "QGAR" est deposé.
- INRIAGforge, SVN, Doxygen, CMake...
- Linux/Windows.

# CMake

- Outil de configuration multiplate-forme.
- Plus simple, plus pratique et plus mantenible que **autotools/autoconf**.
- Native en Linux, Windows, Mac, etc...

Propose encore:

- Une plataforme de testes.
- Génération de paquets (**Debian**, **RPM**, **NSIS**, etc.)

# La bibliothèque de base **QgarLib**

- 100-150 classes.
- Coeur de la plate-forme
- Critères de codage: **performance** et **génericité**.

La bibliothèque de base contient les primitives graphiques, classe pour la communication externe, exceptions, données et fonctions globales, E/S, traitements graphiques, traitements d'image, etc.

# Génericité

- En simplifiant, **Généricité =**
  définir et utiliser du code/objets paramétré(s) par des types.
- **C++** $\Rightarrow$ **templates**.

Dans QGAR?

- On gagné en généralité!
- 1 seule definition pour toutes sortes de types/objets.
- Gain d'écriture, gain du temps, moins code, moins travail...

# Génericité

Plusieurs sortes d'images

- N&B, binaire : short int
- niveaux de gris: long int
- couleurs : 3 x [type]
- gradient : double
- etc...

$\Rightarrow$ **généricité!**
évite opérations, primitives pour chaque type.

# GenImage

L'image est l'Objet de base.
Les caracteristiques de **GenImage**:

- Factorisation des caracteristiques communes à toutes les images.
- Structure de données = 1 tableau de pixels de type parametre.
- Définition des opérateurs (+, -, *, +=, etc.)

**GenImage** est la classe base de tous les images utilisés dans la plate-forme.

- **NiblackBinaryImage**, **CleanedBinaryImage**, **CloseImage**, **DilatedImage**, **GeodesicRecEroImage**, **KanungoBinaryImage**, et beacoup d'autres (34 total!).

# GenImage

```
381  template
382  <
383    class T,
384    template<class> class CheckPolicy =
                                GenImage_NoCheck
385  >
386  class GenImage
387
388    : public CheckPolicy<T>
```

# GenImage - Les bornes d'image

- Une classe pour définir la politique de vérification de la taille de l'image (bornes).
- NoCheck : rien.
- Bound_Check : on verifie.

```
136  template <typename T>
137  class GenImage_BoundCheck
```

- Instanciation :

```
GenImage<unsigned char, TestPolicy> bDupImg(bImg);
```

# GenImage - Constructeurs

```
404  typedef T value_type;

505  template<template <class> class OtherCheckPolicy>
506  GenImage(const GenImage<value_type, OtherCheckPolicy>
             & anImg);
```

- Le paramètre *OtherCheckPolicy* dans le template GenImage, qui est argument du constructeur GenImage, est un **template** aussi.

# GenImage - Constructeurs

```
519  template<class U,template<class> class UCheckPolicy>
520  explicit GenImage(const GenImage<U, UCheckPolicy>
                       & anImg);
```

- Initialization d'une image a partir d'une image avec pixels differents.

# GenImage - Implementation

- Vérification et copie du contenu d'une image.

```
// CONVERSION FROM AN IMAGE HAVING SAME PIXEL TYPE 'T'
// BUT DIFFERENT POLICIES

template <class T, template<class> class CheckPolicy>
template <template<class> class OtherCheckPolicy>
GenImage<T, CheckPolicy>
  ::GenImage(const GenImage<T, OtherCheckPolicy>& anImg)

  : _bytesPerPixel(anImg.bytesPerPixel()),
    _pRefCnt(new int(0)),
    _width(anImg.width() ),
    _height(anImg.height() )
```

# GenImage - Implementation

```
{  // Ensure that passed pixmap does not contradict
  //image invariant.
  checkRange(anImg.pPixMap(), _width, BoundingBox(Point(),
             _width, _height));
  int size = _width * _height;
  _pPixMap = new T[size];
  memcpy(_pPixMap, anImg.pPixMap(), size * sizeof(T)); }
```

- Instanciation:

```
GenImage<float,  TestPolicy> fDupImg(fImg);
```

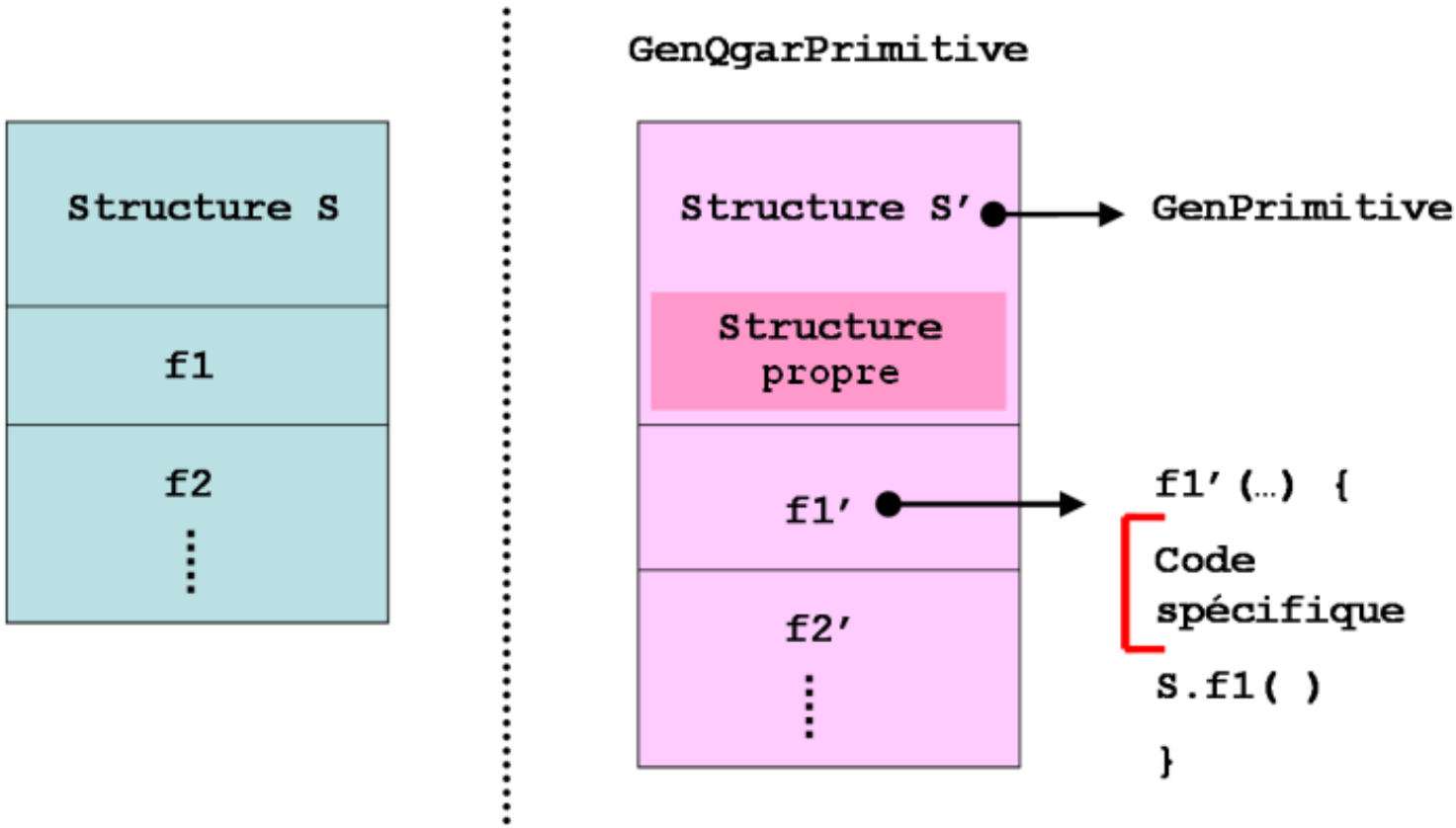
# Primitives

Objets utilisés dans le traitment graphique et d'images.

- Les primitives géométriques: segments de droite, polylines, polygones et arcs de cercle.
- Les primitives QGAR ou primitives graphiques.
  En fait, primitives avec attributes en plus.
  1. Couleur
  2. Épaisseur
  3. Pointillés

# Primitives

## Example d'**Aggregation**



# Composant Connexe

Optimisation

- Espace : structures de données.
- Temps de calculs : algorithmes performants programmées de manière optimisé.

Région connexe de pixels noirs ou blancs.

- Décomposition en composants connexes.
- 2 passes sur l'image : 1 passe d'étiquetage, 1 passe de rétiquetage.

Algorithme **optimisé**.

- Une classe (**ConnectedComponentsImpl**) pour construire les composants connectes.
- Une étiquette est associé a chaque composant $\Rightarrow$ attribut
- Composants stockés dans une arbre.

# Tests unitaires

- Garantie de la fiabilité de l'implementation (dans une certaine mesure).
- Éviter la dégradation du code source.
- Assurer la correction d'une unité du code source.

Tests unitaires en QGAR

- Une classe est la piéce 'atomique' a tester.
- Utilisation d'une outil extérieur: **cppunit**.
- Nouvèlle classe, nouveaux test unitaire correspondant.

Les tests sont exécutés sur toutes les plateformes et aprés chaque modification du code source.

# Chiffres (un peu datés!)

**All experiments performed with Qgar 2.1.1**

**Pentium 866 Mhz, 256 Mb RAM, Linux**

grey level images | binarized images

**MATHEMATICAL MORPHOLOGY**

- average CPU times (in seconds) of 3 experiments
- *only 1 experiment when indicated in italics*

| images | 1024 x 924 | 4752 x 3507 | 19500 x 6000 |
|---|---|---|---|
| linear horizontal dilation | 0.02 | 0.77 | 21.85 |
| | 0.04 | 0.23 | 2.35 |
| linear vertical dilation | 0.02 | 1.08 | 43.23 |
| | 0.04 | 0.37 | 14.56 |
| linear diagonal dilation | 0.02 | 1.09 | 31.80 |
| | 0.04 | 0.42 | 5.54 |
| dilation + erosion | 0.07 | 1.72 | 49.75 |
| | 0.07 | 0.69 | 20.00 |
| opening + closing | 0.13 | 3.20 | *83.61* |
| | 0.10 | 1.77 | 40.62 |

# Pas encore prêt?

L'environnement de recherche:

- Je code, tu codes, il code...
- Je me dépêche, tu ne testes pas, il n'aime pas OO...
- Je propose un nouvel algorithme, tu augmentes la performance d'un autre, il ajoute une nouvèlle classe d'images...
- Je n'écris pas la doc, tu commentes le code en hollandais, il... Oú-est-il? Il a déjà fini sa thèse?

Algorithmes, implementátions, classes sont ajoutés et sont disponibles à tous.
**Mais** a quel prix? $\Rightarrow$ **QgarDevLib**

# Pas encore prêt?

**Donc**, il faut passer à la bibliothèque QGAR. Pour le faire, il faut:

- Coder les tests unitaires.
- Suivre le standard de codage QGAR.
- Documentation complète au format doxygen.

Dans la mesure do possible, il faut avoir le minimum possible des fonctions dans la bibliothèque de développement et plus dans la bibliothèque de base.

# Applications QGAR

- Executables independants.
- Mêmes standard de programmation.
- Utilisation en ligne de commande.
- Utilisation dans l'interface graphique.
- Skeletonization, edge detection, degradation, text extraction, vectorization, binarization, etc...
- Pas de tests unitaires.

1 application = 1 fichier de description XML + 1 executable

# Chiffres (un peu datés!)

Unknown processor, Unix

Unknown processor, 196 Mb RAM, Windows

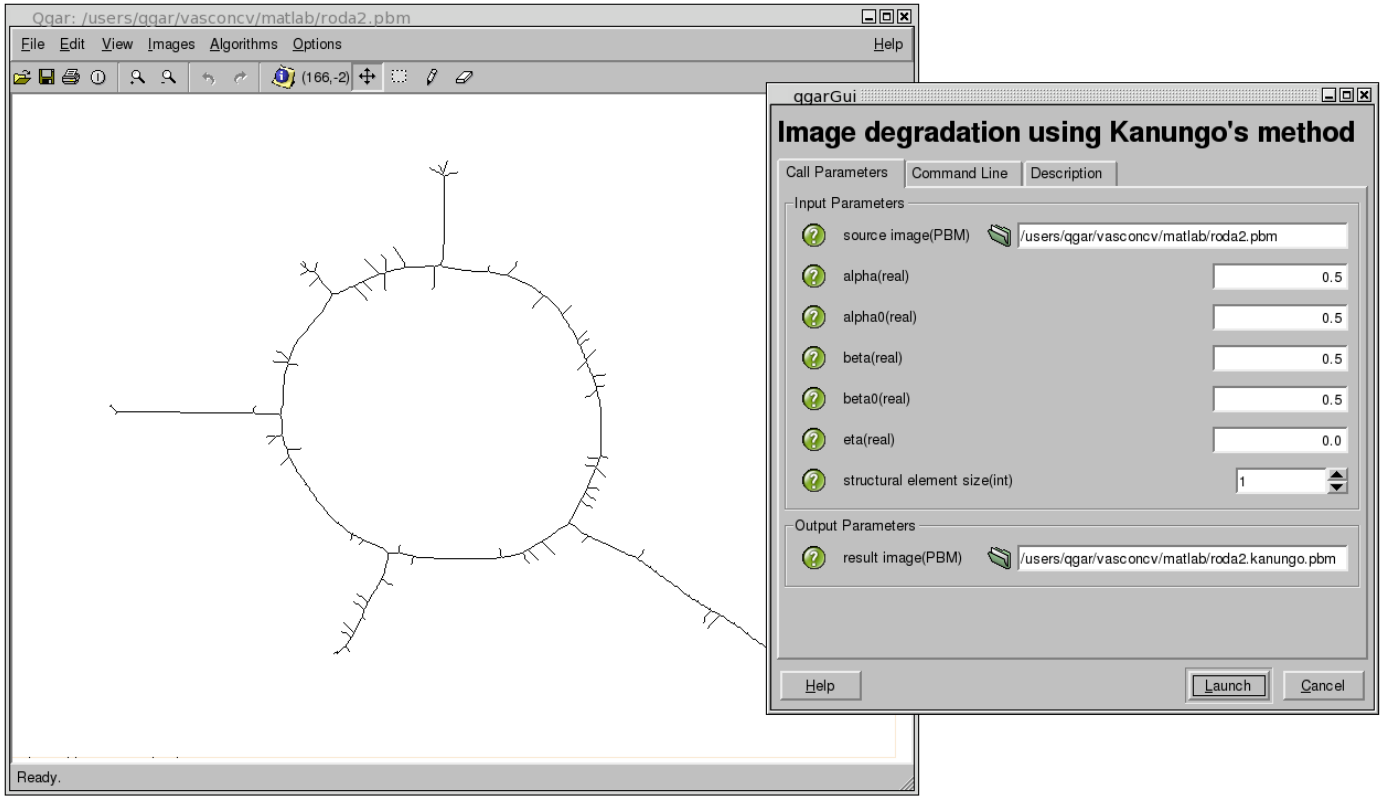
Courtesy of EDF (Electricité de France)

| images | 4480 x 8666 | 10784 x 11759 | 10784 x 12839 | 8660 x 2384 | 8679 x 2384 | 8760 x 2352 |
|---|---|---|---|---|---|---|
| text/graphic separation | 522 | 4674 | 1075 | 830 | 863 | 1860 |
| text extraction | 142 | 1342 | 150 | 198 | 253 | 472 |
| **total CPU seconds** | 664 *11.1mn* | 6016 *100.3mn* | 1225 *20.4mn* | 1028 *17.1mn* | 1116 *18.6mn* | 2332 *38.9mn* |
| text/graphic separation | 60 | | | 107 | 116 | 263 |
| text extraction | 12 | | | 21 | 28 | 66 |
| **total CPU seconds** | 72 | | | 128 | 144 | 329 |

# QgarGUI

- Interface graphique pour utiliser les applications.
- Visualisation des resultats et corrections.
- Qt + C++.
- 1 nouvelle application $\Rightarrow$ écrire la description ensuite elle est appelable depuis l'interface.

# QgarGUI


Qgar: /users/qgar/vasconcv/matlab/roda2.pbm
File Edit View Images Algorithms Options Help
(166,-2)
Ready.
qgarGui
Image degradation using Kanungo's method
Call Parameters
Command Line
Description
Input Parameters
source image(PBM)
/users/qgar/vasconcv/matlab/roda2.pbm
alpha(real)
0.5
alpha0(real)
0.5
beta(real)
0.5
beta0(real)
0.5
eta(real)
0.0
structural element size(int)
1
Output Parameters
result image(PBM)
/users/qgar/vasconcv/matlab/roda2.kanungo.pbm
Help
Launch
Cancel

# Une plate-forme stable

- Utilisé pour toute l'équipe QGAR ainsi que pour les partenaires.
- Disponible également en Linux et Windows. (Mac en théorie, pas encore testé)
- Documentation **detaillée** pour dévéloppeurs et utilisateurs.

# Échancier 2008

- Passage à Qt4.
- Standartization et documentation du code de la bibliothèque **QgarDevLib**.
- Automatisation des testes unitaires.

Souhaites...

- Utilisation de la norme SVG comme format privilégié pour les échanges.

# Description de la passage à Qt4

- Grande évolution par rapport la version 3.
- Important changement en classes d'affichage de **Pixmaps**.
- Assez grande... pas retro compatible.

Méthode *ad-hoc*:

1. Simplification d'un lot minimum de classes (commenataire!).
2. Codage Qt3 $\Rightarrow$ Qt4.
3. Compilations en Linux et Windows.
4. Tests de fonctionalités.