MORPH-M

Une bibliothèque extensible de traitement morphologique d'images

Raffi Enficiaud et Romain Lerallut
Thibauld Nion et Thomas Retornaz

Plan

- Introduction
- Briques
- Interface Python
- Outils collaboratifs et AQ
- Conclusion

Introduction

- Lancé le 21 janvier 2003
- Héritier d'Xlim3D
- Étendre et améliorer Xlim:
  - fournir des outils flexibles pour la conception d'algorithmes
  - permettre une ré-utilisabilité maximale
  - grande abstraction

Quelques chiffres

- Cœur:
  - 129 389 lignes de code (95% en C++)
  - 33 personne-années
  - Coût estimé de développement: 3 M€
- addons
  - 96 425 lignes
  - 2M€
- users, users_emeritus ...

Estimations par sloccount (de D. Wheeler) et le modèle de coûts COCOMO
Le calcul des lignes de code est « consolidé » et bien inférieur au décompte direct que l'on peut faire à la main.

Utilisation actuelle

- la majorité des thésards du CMM à Fontainebleau
- TDs de morphologie pour optionnaires (depuis 2005)
- “Simulation of random media” (SIMEA)
- projets PACA, Geodesis, “Surface Idéale”, Clovis, PICS, EADS, Cosmeca, Collège de France ...
- labos extérieurs: tortuosité avec Evry , ARSA, Michelin

Concepts fondateurs

BESOINS

Concepts fondateurs

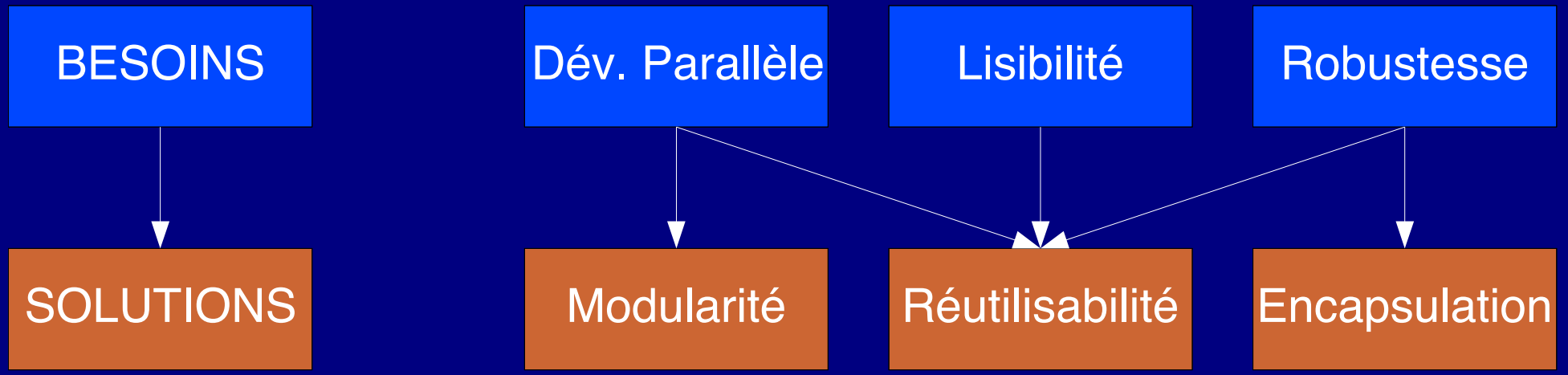
BESOINS

Dév. Parallèle

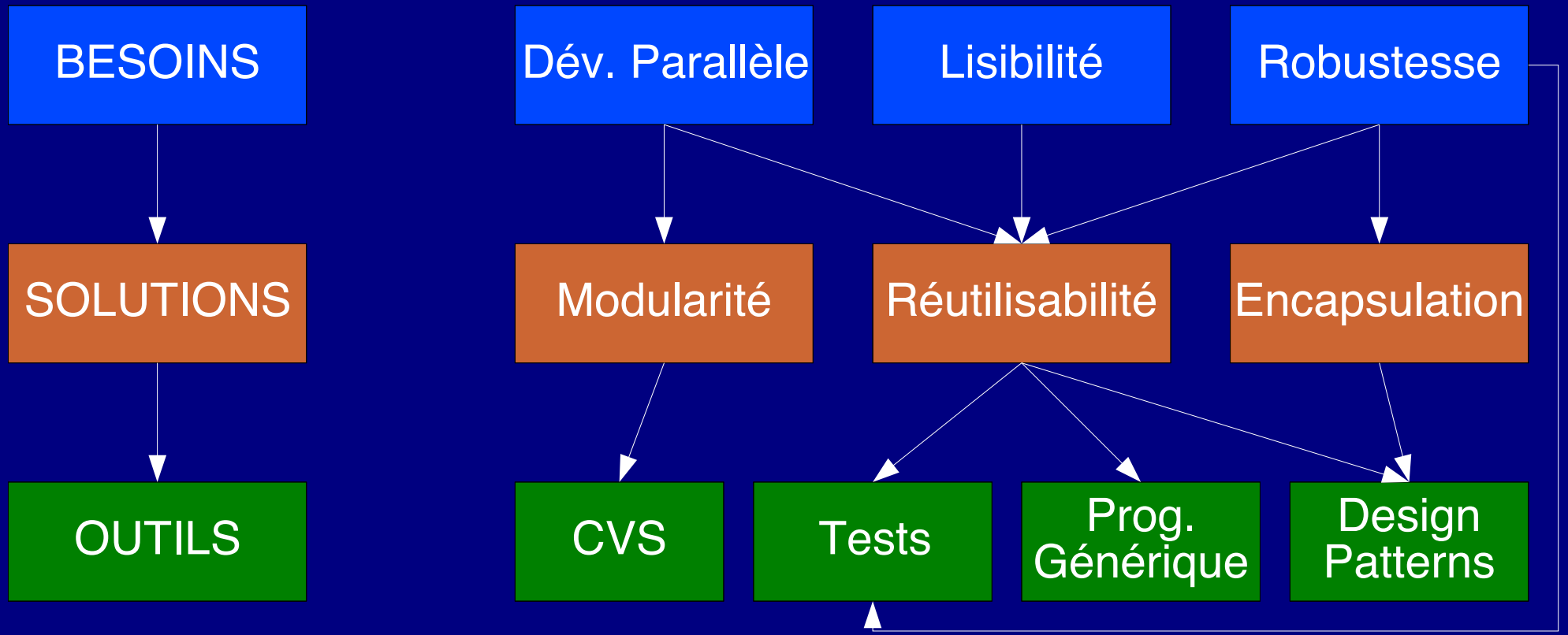
Lisibilité

Robustesse

Concepts fondateurs



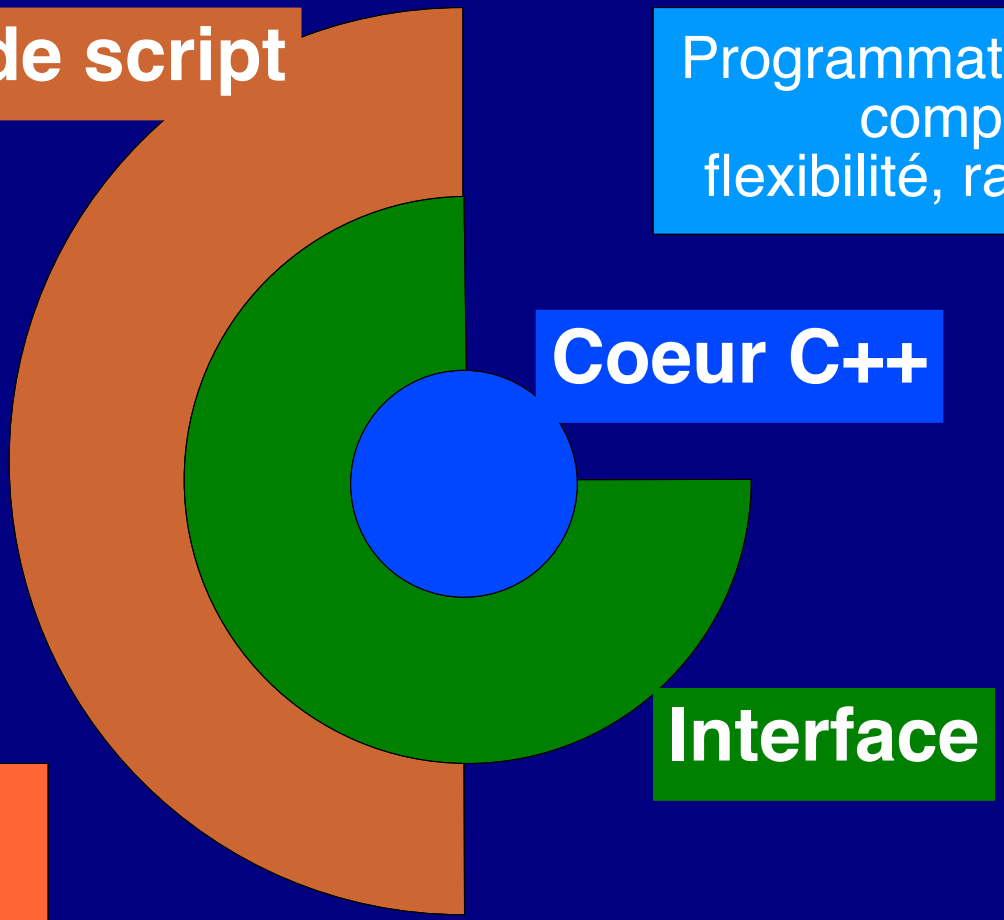
Concepts fondateurs



Portabilité

- PCs « standards », 32 bits
  - MS Windows et Linux
- Apple Macintosh 32 bits (G4) et 64 bits (G5)
  - Apple OS X (10.4 *Tiger*) et Linux/PPC
- Serveur AMD 64 bits
  - Linux 64 bits
- Serveur XPRO 64 bits
  - Support expérimental (Note: limitation = msvc 2005)

Organisation en couches



Niveaux de Distribution

- **code source complet** (CMM seulement)
  - C++ bas niveau (algorithmes optimisés)
- **interface C++** (ARSA, Clovis)
  - coder en C++ simple (fonctionnalités type Xlim)
- **Python seul** (TDs)
  - coder en Python (prototypage d'algos)
- **Librairie/executable spécifique** (PACA, Segmentator)
  - utilisation seulement dans le cadre prévu

Plan

- Introduction
- Briques
  - abstraction
  - itérateurs et opérateurs
  - voisinages
  - contrôleurs
- Interface Python
- Outils collaboratifs et AQ
- Conclusion

Notion fondamentale:

Abstraction des types de données

Une image est un ensemble de valeurs

- loi de composition interne: « + »

  si « x + y » est défini, alors on peut écrire « Im1 + Im2 »

- ordre: « < »

  si « x < y » est défini, alors on peut :

  - définir un treillis => érosions, dilatation, reconstructions, LPE,...
  - utiliser nos F.A.H => implémentations optimales

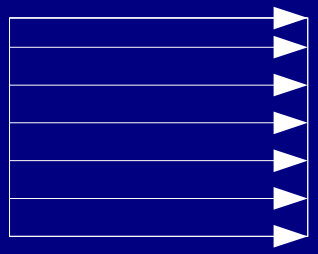
- distance: « d »

  si « d(x , y) » est défini, alors on peut faire des lambda-labélisations, croissances de région, etc.

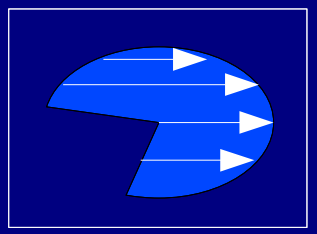
Le **compilateur** se charge des détails

Itérateurs

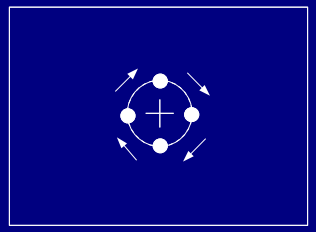
« Pour tous les éléments de X, faire ... »



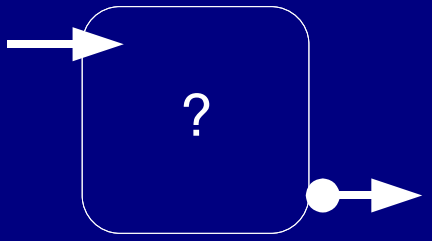
Itérateur simple



Itérateur avec masque



Itérateur sur voisinage



Itérateur complexe

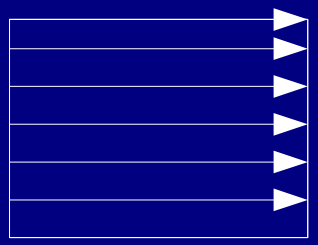
Opérateurs

- permet de séparer l'opération du type de données sur lequel on l'applique:

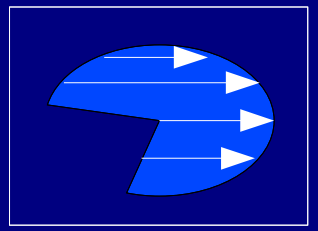
- utiles quand l'opération peut être utilisée ailleurs
- améliore la lisibilité du code

Opérateurs (2)

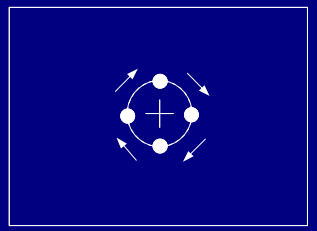
Un même opérateur, couplé à des itérateurs différents produit des fonctions différentes.



Itérateur simple + Max = **ImMaximum**



Itérateur avec masque + Max = **ImMaximumOnRegion**

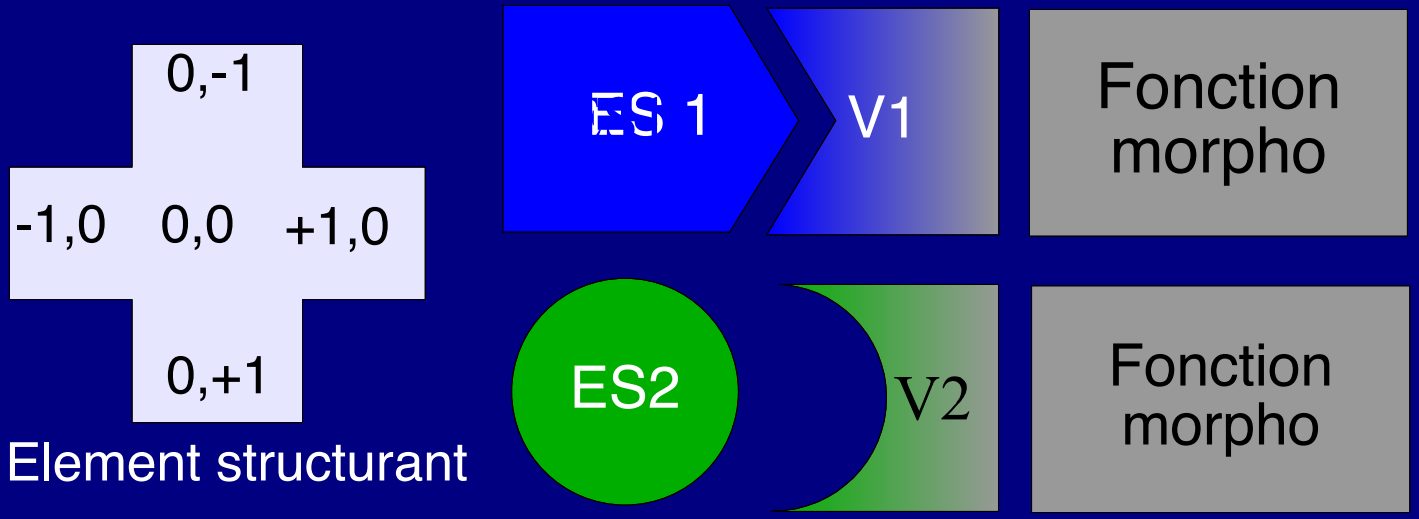


Itérateur sur voisinage + Max = une étape de **ImDilate**

Voisinages

« Pour tous les voisins de P, faire... »

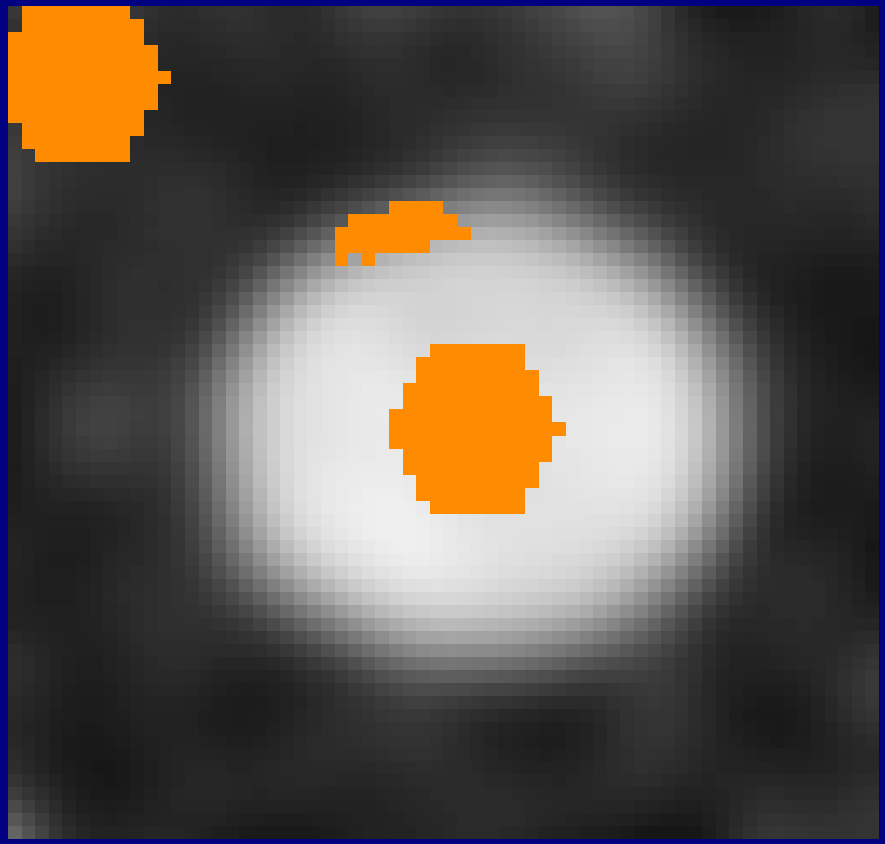
- Fournit une interface unique :



- Pour un nouvel ES, il suffit de créer le voisinage correspondant pour se connecter à **toutes** les fonctions morphologiques compatibles.

Voisinages, suite

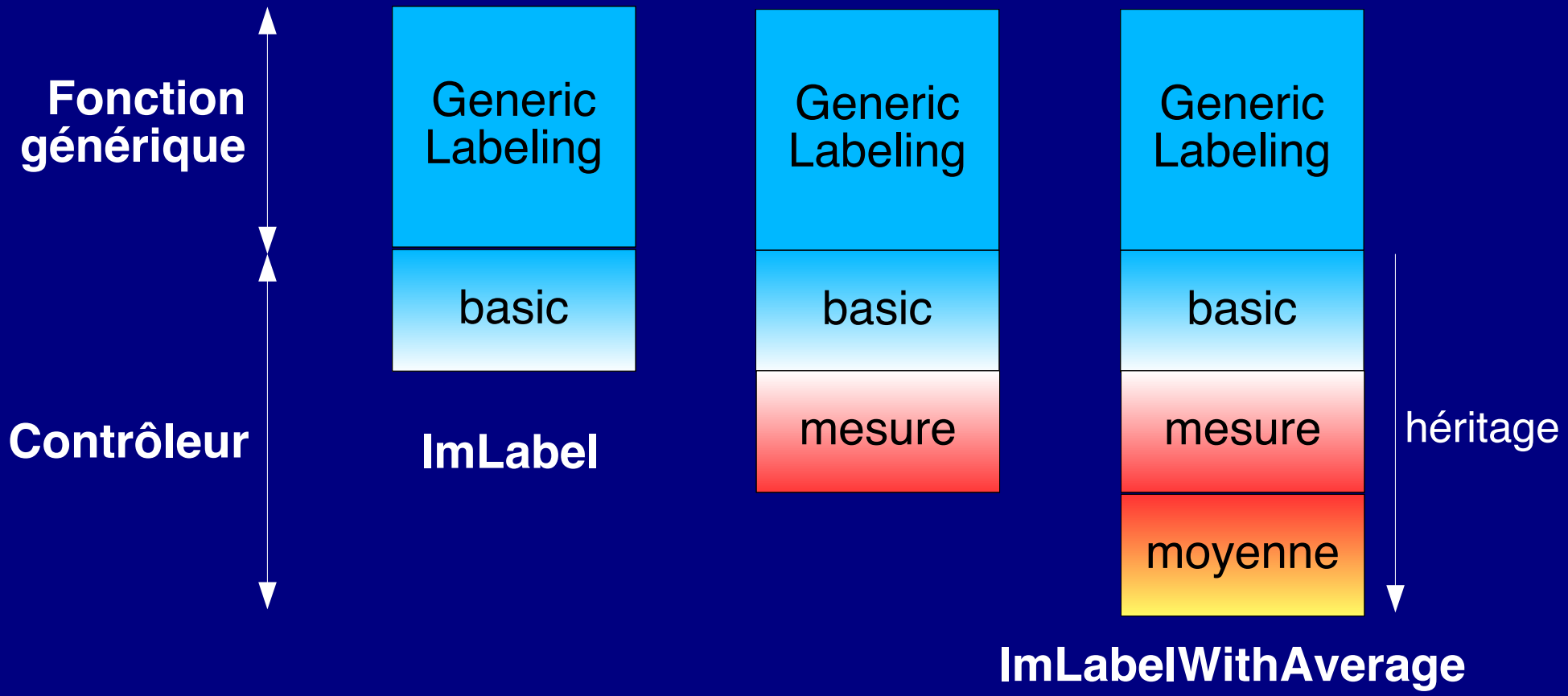
La séparation en éléments structurants et voisinages a permis d'implémenter *toute* la morphologie sur les « amibes » en une matinée (de Romain).



Idem pour les « MotionSE » de N. Laveau

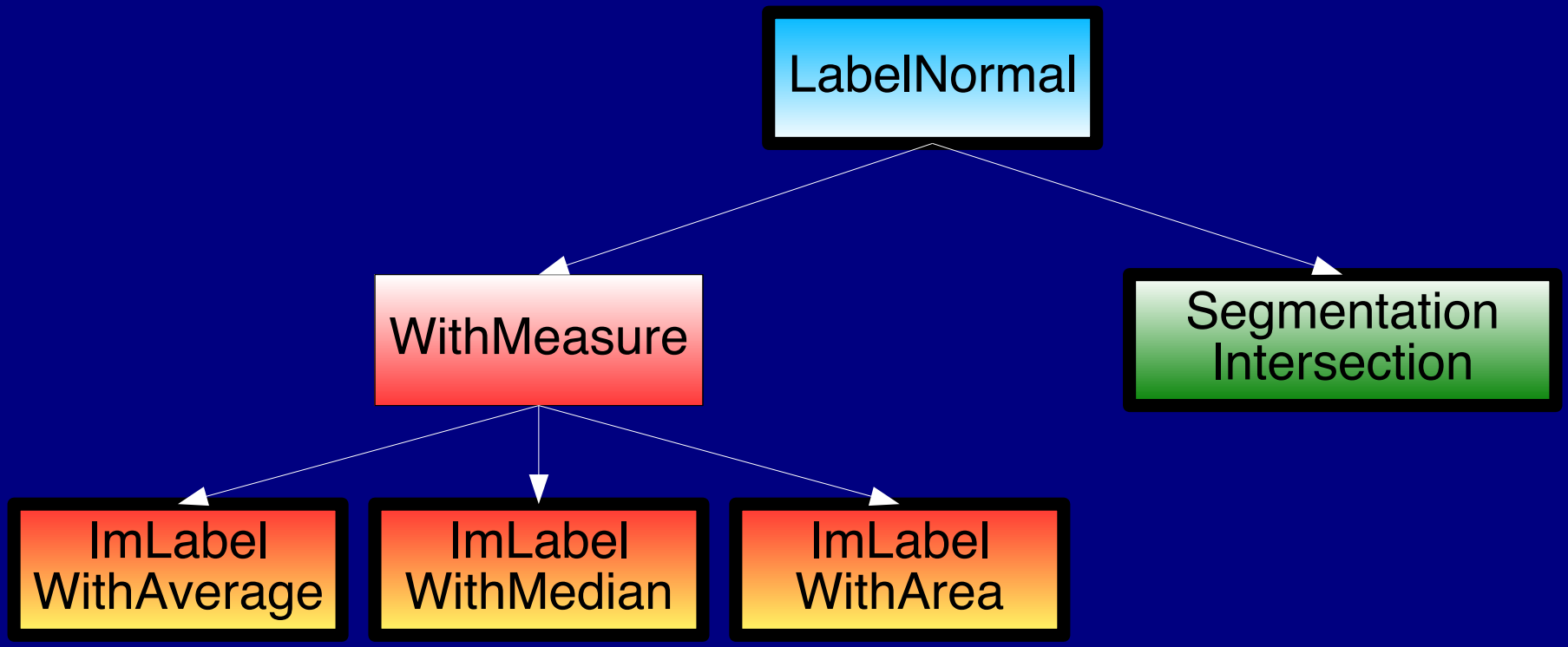
Contrôleurs

- utilisés avec une fonction générique pour créer des familles d'algorithmes semblables.



Contrôleurs

- Les relations d'héritage entre les contrôleurs exposent les relations conceptuelles entre les algorithmes:



Contrôleurs: les 7 erreurs

Ces deux algorithmes sont-ils équivalents ?

```
current=1;
for(offs=0;offs<in.size();++offs){
    if(in[offs] != 0){
        if(tmp[offs]!= DONE){
            tmp[offs]=DONE;
            queue.push(offs);

            while(! queue.empty()){
                o0=queue.pop();
                out[o0]=current;
                for
o1=0;o1<numNeighb;++o1){

tmpOf=o0+neighbs[o1];
                    if in[tmpOf]!=0 &&
tmp[tmpOf]!= DONE){

queue.push(tmpOf);

tmp[tmpOf]=DONE;}}}}
            current++;
}}
```

```
for(curLabel=1, x=0;x<xsize;++x){
for(y=0;y<ysize;++y){
for(z=0;z<zsize;++z){
    offs=getOffset(x,y,z)
    if(*(workAddr+offs) != DONE){
        *(workAddr+offs)=DONE;
        queue.push(offs);
        do{
            o0=queue.pop();
            *(outAddr+o0)=curLabel
            for(o1=0;o1<numNeighb;++o1){
                if (*(inAddr+neighbs[o1])
== *(inAddr+o0) &&
*(workAddr+o1)!=DONE){
                    queue.push(o1);
                    *(workAddr+o1)=DONE;}
            }
        }while(!queue.empty());
        curLabel++;
    }
}}} // for x,y,z
```

Contrôleurs: les 7 erreurs, suite

Ces deux classes ont-elles le même comportement ?

```
class CGauche
{
 bool pixelCanBeLabelled(p)
 {
   return p!=0;
 }

 bool
pixelsAreConnected(p,q)
 {
   return q!=0;
 }
};
```

```
class CDroite
{
 bool pixelCanBeLabelled(p)
 {
  return true;
 }

 bool pixelsAreConnected(p,q)
 {
  return p==q;
 }
};
```

Contrôleurs: les 7 erreurs, suite

Ces deux classes ont-elles le même comportement ?

```
class CGauche
{
 bool pixelCanBeLabelled(p)
 {
   return p!=0;
 }

 bool
pixelsAreConnected(p,q)
 {
   return q!=0;
 }
};
```

```
class CDroite
{
 bool pixelCanBeLabelled(p)
 {
  return true;
 }

 bool pixelsAreConnected(p,q)
 {
  return p==q;
 }
};
```

- **CGauche** est utilisée pour implémenter une labellisation de blobs sur fond noir
- **CDroite** est utilisée pour labelliser des zones plates.

Briques: conclusion

- Outils haut-niveau (*« pour tout x dans X, ... »*)
- Faible duplication du code (factorisation)
  - rapidité de développement
  - lisibilité
  - bugs
- Fiabilité des couches de base (confort, sécurité)
- Facilités de prototypage (ajout d'une variante)

Inédits

- Images hyperspectrales
- Images à n dimensions
- Treillis lexicographiques
- Grosses images (64 bits, plusieurs Go)
- Amibes, ESM, Amibes+ESM
- ...

Disponibles *maintenant*

Plan

- Introduction
- Briques
- Interface Python
  - simplicité
  - prototypage
  - transparence
- Outils collaboratifs et AQ
- Conclusion

Interface Python

- Python
  - langage orienté objet (comme C++)
  - dynamique (comme Lisp)
  - populaire (comme Java)
  - expressif (comme personne ! Développement DWIM)
- Interface avec Morph-M
  - développement rapide (expressif et « sécurisé »)
  - couche mince au-dessus de C++ (faible duplication)

Interface Python: librairies externes

- Traitement numérique:
  - R , Scientific Python , FFTW... (Thibauld, Raffi)
  - Matlab
- Visualisation – Interface Graphique:
  - Gnuplot
  - VTK (Etienne / Jean)
  - toolkits graphiques (Thibauld , Raffi)
- Réseau (Raffi)

Interface Python: simplicité

Code simple:

```
im    = fileRead( « foreman.png » )
imOut = ImCreateSame( im )

ImDilate( im, SquareSE, imOut )
```

Gestion complètement automatique de la mémoire:

```
while 1:
  im = fileRead( « foreman.png » )
```

Interface Python: prototypage

Facilités pour utiliser du code Python:

```
# complémentaire à 255:
 def complem255(x):
   return 255 - x

 ImUnaryOperation( imIn, complem255, imOut )
```

Beaucoup d'opérations élémentaires:

```
# une dilatation

 ImNeighborhoodOperation( imIn, SE, max, imOut )
```

Ces opérations reflètent (noms identiques) les fonctions génériques disponibles dans le coeur C++/template (facilité de portage Python -> C++).

Interface Python: transparence

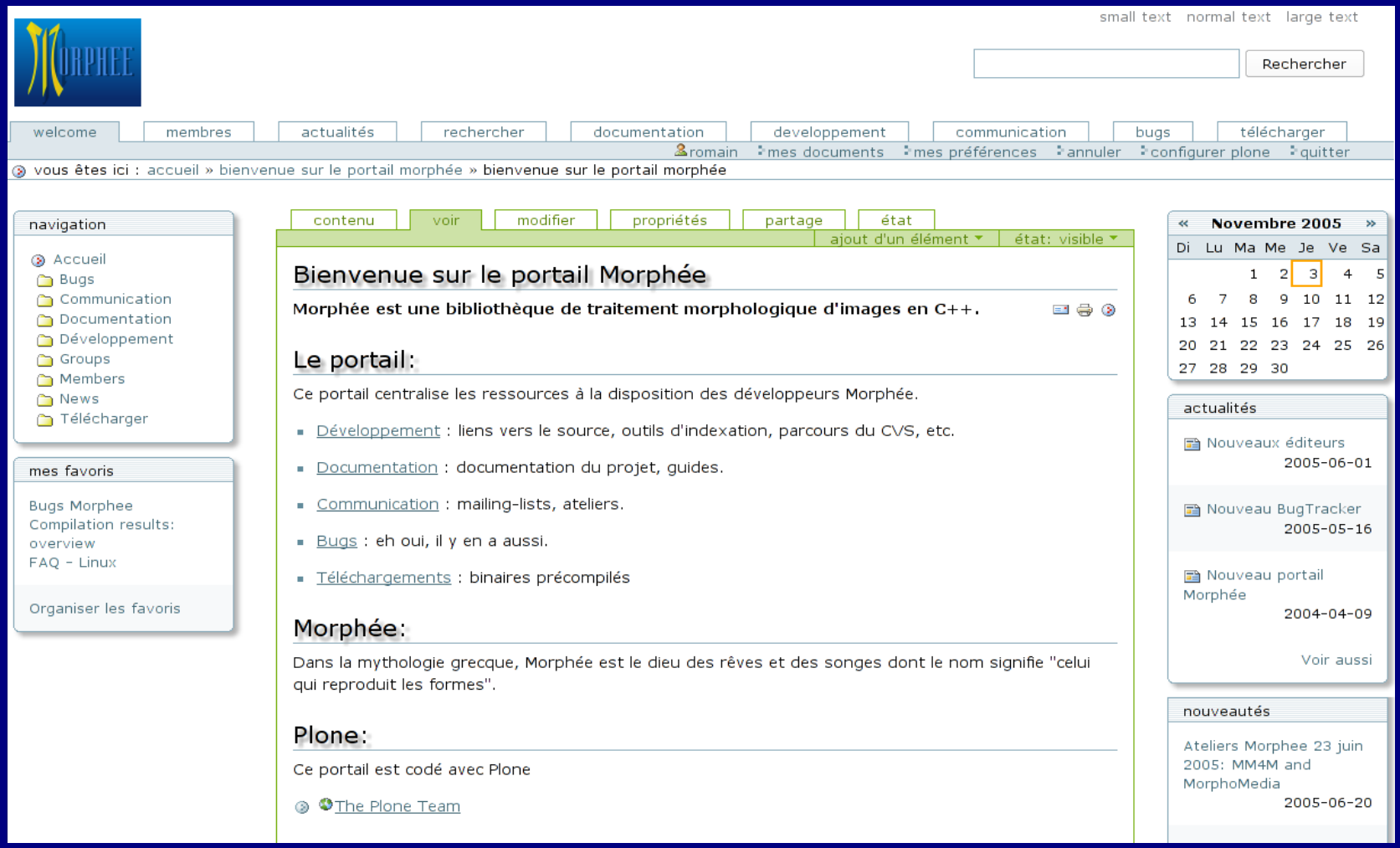
- Squelettes de fonction disponibles en Python:
  - opérations unitaires (complémentaire, conversions)
  - opérations binaires (arithmétique,...)
  - opérations sur voisinages (érosions, etc)
- Fonctions génériques:
  - labellisations
  - inondations

Plan

- Introduction
- Briques
- Interface Python
- Outils collaboratifs et Assurance Qualité
  - portail
  - CVS
  - listes de diffusion
  - tests
- Conclusion

Portail MorphM

- http://rhodes.ensmp.fr/morphee



Documentation: fonctions

Générée automatiquement à partir du code

Morphee User Documentation

Overview | Modules | Classes (annotated) | Namespaces | Search for

Estimation

[Motion]

Functions

__Mmot RES_C morphee::motion::BlockMatchingEstimation (const ImageInterface *imCurrent, const ImageInterface *imReference, coord_t precision, coord_t density, UINT32 allowedBlockSizes, coord_t maxSearchRange, ImageInterface *imMotion)

Blockmatching estimation with various possible blocksize and no constraint.

__Mmot bool morphee::motion::accept_factor (coord_t factor)

Returns true if factor is a power of 2.

Detailed Description

Functions for motion estimation. Currently only block-matching.

Function Documentation

```
__Mmot RES_C morphee::motion::BlockMatchingEstimation( const ImageInterface * imCurrent,
                                                       const ImageInterface * imReference,
                                                       coord_t                precision,
                                                       coord_t                density,
                                                       UINT32                 allowedBlockSizes,
                                                       coord_t                maxSearchRange,
                                                       ImageInterface *       imMotion
                                                     )
```

Documentation: exemples

Examples

Simple labellings

```
using namespace morphee;

// Labels (1,2,3, ...) the blobs of imIn on a black background
morphoBase::ImLabel(imIn, selement::neighborsSquare2D, imLab);

// Labels (1,2,3, ...) all the flat zones of imIn
morphoBase::ImLabelFlatZones(imIn, selement::neighborsHex2D, imLab);

// Labels (1,2,3, ...) all the "lambda" flat zones  (Treshold distance = Lambda)  of imIn
morphoBase::ImLabelLambdaFlatZones(imIn,Lambda,selement::neighborsHex2D, imLab);
```

Labellings with other values

The following functions don't label with an increasing counter (1,2,3,...) but instead they compute the shape of the region, compute a value (area, average value) and affect it to the entire component.

The following code computes the area of each blob, and sets it as labelling value in imLab:

```
using namespace morphee;
morphoBase::ImLabelWithArea(imIn, selement::neighborsHex2D, imLab);
```

However, the value can also be computed on another image:

```
// Using the same labelling as above (blobs on black background)
// but the label value is now computed on imValues
using namespace morphee;
morphoBase::ImLabelWithAverage(imIn, imValues, selement::neighborsSquare2D, imLab);
```

Documentation: structures

Générée automatiquement à partir du code

Morphee Developer Documentation

Overview | Modules | Class Hierarchy | Classes | Classes (annotated) | Members | Namespaces | Search for

morphee::morphoBase::s_LabellingMeasureAverage

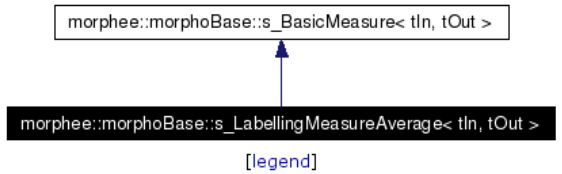
morphee::morphoBase::s_LabellingMeasureAverage< tIn, tOut > Struct Template Reference

[Labelling template operators]

Labelling operator for average measure within the connected components. More...

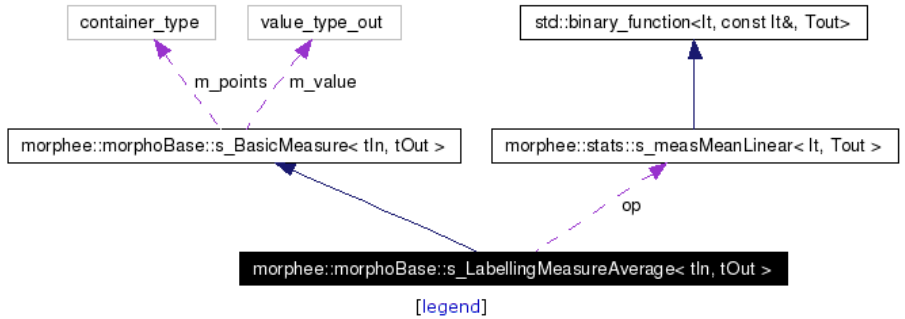
#include <morphoLabel2_T.hpp>

Inheritance diagram for morphee::morphoBase::s_LabellingMeasureAverage< tIn, tOut >:



[legend]

Collaboration diagram for morphee::morphoBase::s_LabellingMeasureAverage< tIn, tOut >:



[legend]

List of all members.

CVS

- outil de gestion de code en collaboration
- stocke *toutes* les versions du programme
- stocke les commentaires associés à *chaque* modification
- permet de revenir en arrière, comparer, etc
- existe sous toutes les plateformes

CVS

Interface web:

[Development] / morphee / morphee / stats / include / private

Development: morphee/morphee/stats/include/private

Current directory: **[Development] / morphee / morphee / stats / include / private**

Files shown: **15**

File	Rev.	Age	Author	Last log entry
Attic/ [show contents]				
statsBasic_T.hpp	1.12	2 months	lerallut	C'est moche, blabla, mais c'est bien pratique... J'en profite pendant que Raf a ...
statsBasic_T_impl.hpp	1.16	2 months	lerallut	C'est moche, blabla, mais c'est bien pratique... J'en profite pendant que Raf a ...
statsEstimation_T.hpp	1.20	7 days	nion	Remplacement de qq numeric limits.
statsLATools_T.hpp	1.17	2 months	enficiau	static_cast
statsLA_LU_T.hpp	1.11	6 days	enficiau	type de comparaison (cette fois j'ai fait gaffe)
statsLA_SVD_T.hpp	1.6	3 months	lerallut	Wow, il est beau celui-là, bien joué Romain.
statsLA_Tools2_T.hpp	1.1	6 months	lerallut	Stockage temporaire (*) d'une implémentation de stats::Vector et stats::Matrix b...
statsLA_eigen_T.hpp	1.3	6 months	lerallut	MOAC2: Déplacement des fichiers d'en-tête privés Belle perf', je sais pas si on...
statsLA_eigen_T_impl.hpp	1.2	21 months	lerallut	MOAC: The Mother Of All Commits Fusion de la branche Devel_1 dans la branche pr...
statsManipulation_T.hpp	1.8	7 days	nion	Remplacement de qq numeric limits.
statsMeanShift_T.hpp	1.1	11 months	lerallut	Recherche du mode le meilleur et le plus proche dans une distribution (dimension...
statsMeanShift_T_impl.hpp	1.4	8 months	enficiau	Suppression de code inatteignable/mort Suppression de paramètres inutilisés
statsMeasure_T.hpp	1.49	7 days	nion	Remplacement de qq numeric limits.
statsNumberUtilities_T.hpp	1.1	7 months	lerallut	Utilisation amusante (et utile !) des templates. A enrichir.
statsNumberUtilities_T_impl.hpp	1.1	7 months	lerallut	Utilisation amusante (et utile !) des templates. A enrichir.

Show files using tag: Select Branch

Show

Download tarball

CVS - suivi des modifs

Chaque modification enregistrée est accompagnée du nom, date et d'un commentaire:

« petites optimisations

ajout de qq operateurs: *,/,*=,/= de vecteur avec scalaires (idem pr matrices)

ajout d'une factory de matrice "identité" »

```
                unsigned int i;
                for(i=0;i<this->size();++i)
                        (*this)[i]-=valIn;
                return *this;
        }
        const Vector operator-() const
        {
                Vector out(size());
                unsigned int i;
                for(i=0;i<this->size();++i)
                        out[i]=-(*this)[i];
                return out;
        }
        bool operator==(const Vector&vIn)const
        {
                unsigned int i;
                for(i=0;i<this->size();++i)
                        if(vIn[i]!=operator[](i))
                                return false;
                return true;

        }
        inline bool operator!=(const Vector&vIn)const
        {
                return ! this->operator==(vIn);
        }
        //friend std::ostream& operator<< (std::ostream&,const Vector& );
}; // class Vector
```

```
                unsigned int i;
                unsigned int sz=this->size();
                for(i=0;i<sz;++i)
                        (*this)[i]-=valIn;
                return *this;
        }
        const Vector operator-() const
        {
                unsigned int sz=this->size();
                Vector out(sz);
                unsigned int i;
                for(i=0;i<sz;++i)
                        out[i]=-(*this)[i];
                return out;
        }
        bool operator==(const Vector&vIn)const
        {
                unsigned int i;
                unsigned int sz=this->size();
                for(i=0;i<sz;++i)
                        if(vIn[i]!=operator[](i))
                                return false;
                return true;

        }
        inline bool operator!=(const Vector&vIn)const
        {
                return ! this->operator==(vIn);
        }
        Vector& operator*=( const T r)
        {
                unsigned int i;
                unsigned int sz=this->size();
                for(i=0;i<sz;++i)
                        (*this)[i]*=r;
                return *this;
        }
        Vector& operator/=( const T r)
        {
                unsigned int i;
                unsigned int sz=this->size();
                for(i=0;i<sz;++i)
                        (*this)[i]/=r;
                return *this;
        }
        //friend std::ostream& operator<< (std::ostream&,const Vector& );
}; // class Vector
```

CVS - suivi des modifs

Pour chaque ligne, on connait l'auteur de la dernière modification:

```
160 :                                                                   return false;
161 :                                                           return true;
162 : lerallut  1.2
163 : lerallut 1.11                                     }
164 :                                                   inline bool operator!=(const Vector&vIn)const
165 :                                                   {
166 :                                                           return ! this->operator==(vIn);
167 :                                                   }
168 :                                                   Vector& operator*=( const T r)
169 :                                                   {
170 :                                                           unsigned int i;
171 :                                                           unsigned int sz=static_cast<unsigned int>(this->size());
172 :                                                           for(i=0;i<sz;++i)
173 :                                                                   (*this)[i]*=r;
174 :                                                           return *this;
175 :                                                   }
176 :                                                   Vector& operator/=( const T r)
177 :                                                   {
178 :                                                           unsigned int i;
179 :                                                           unsigned int sz=static_cast<unsigned int>(this->size());
180 :                                                           for(i=0;i<sz;++i)
181 :                                                                   (*this)[i]/=r;
182 :                                                           return *this;
183 :                                                   }
184 :                                                   //friend std::ostream& operator<< (std::ostream&,const Vector& );
185 :                                   }; // class Vector
186 : lerallut  1.2
187 : enficiau  1.4                  template<typename T>
188 :                                   const Vector<T> operator+(const Vector<T>&lhs, const Vector<T>&rhs)
189 : lerallut 1.11                    {
190 :                                           return Vector<T>(lhs)+=rhs;
191 :                                   }
```

Listes de diffusion

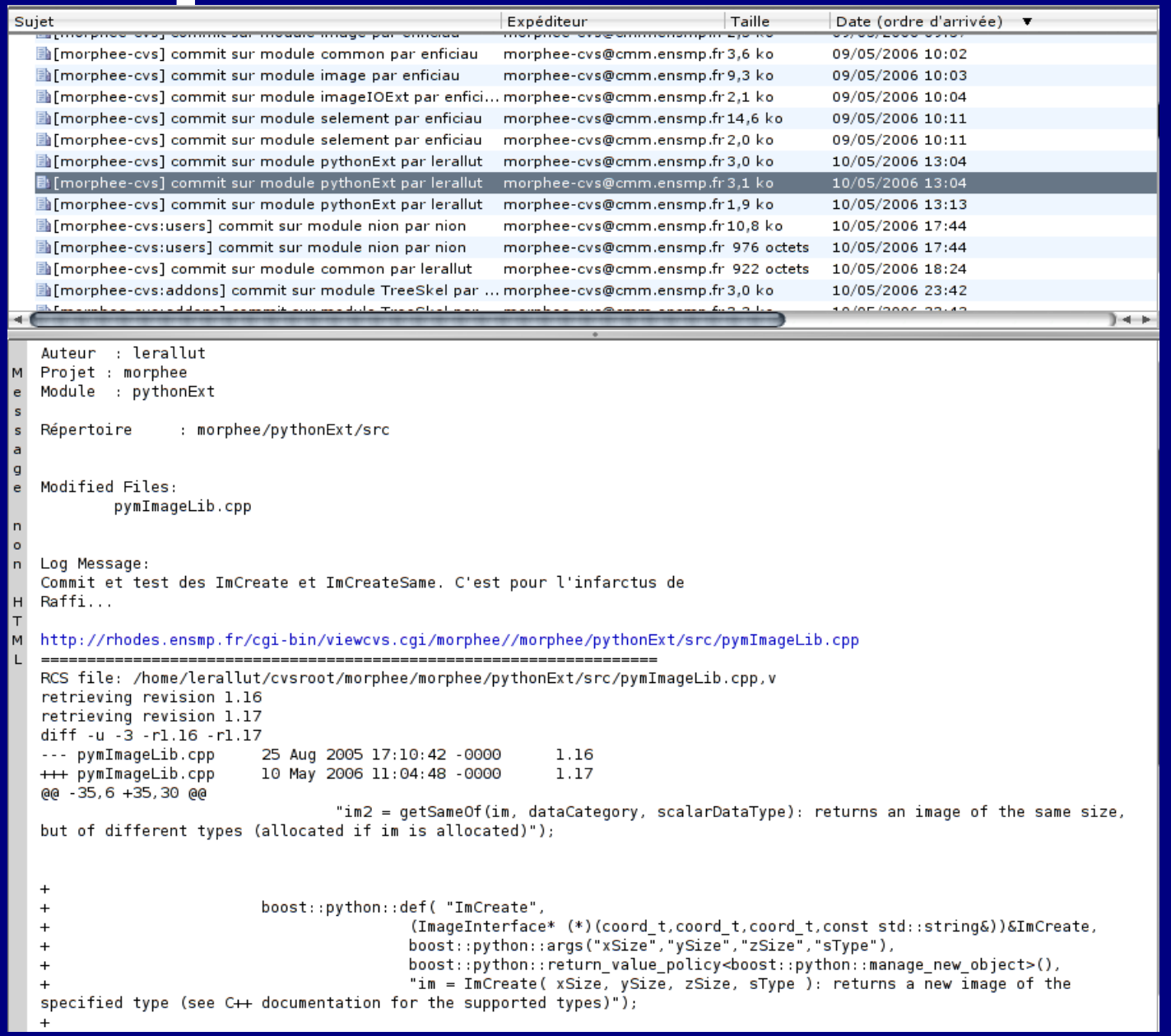
- morphm-users@cmm.ensmp.fr
  - annonces diverses, sondages, avis de tempêtes
  - Résultats des compilations nocturnes
- morphee-cvs@cmm.ensmp.fr
  - copie de toutes les modifications du code
  - favorise la synergie entre développeurs
  - haut débit (faire des filtres !)
- morphm-dev@cmm.ensmp.fr
  - Intermédiaires : discussions, échanges

Liste de diffusion morphee-cvs

Auteur

Fichiers Message

Différences



Sujet	Expéditeur	Taille	Date (ordre d'arrivée)
[morphee-cvs] commit sur module common par enficiau	morphee-cvs@cmm.ensmp.fr	3,6 ko	09/05/2006 10:02
[morphee-cvs] commit sur module image par enficiau	morphee-cvs@cmm.ensmp.fr	9,3 ko	09/05/2006 10:03
[morphee-cvs] commit sur module imageIOExt par enfici...	morphee-cvs@cmm.ensmp.fr	2,1 ko	09/05/2006 10:04
[morphee-cvs] commit sur module selement par enficiau	morphee-cvs@cmm.ensmp.fr	14,6 ko	09/05/2006 10:11
[morphee-cvs] commit sur module selement par enficiau	morphee-cvs@cmm.ensmp.fr	2,0 ko	09/05/2006 10:11
[morphee-cvs] commit sur module pythonExt par lerallut	morphee-cvs@cmm.ensmp.fr	3,0 ko	10/05/2006 13:04
[morphee-cvs] commit sur module pythonExt par lerallut	morphee-cvs@cmm.ensmp.fr	3,1 ko	10/05/2006 13:04
[morphee-cvs] commit sur module pythonExt par lerallut	morphee-cvs@cmm.ensmp.fr	1,9 ko	10/05/2006 13:13
[morphee-cvs:users] commit sur module nion par nion	morphee-cvs@cmm.ensmp.fr	10,8 ko	10/05/2006 17:44
[morphee-cvs:users] commit sur module nion par nion	morphee-cvs@cmm.ensmp.fr	976 octets	10/05/2006 17:44
[morphee-cvs] commit sur module common par lerallut	morphee-cvs@cmm.ensmp.fr	922 octets	10/05/2006 18:24
[morphee-cvs:addons] commit sur module TreeSkel par ...	morphee-cvs@cmm.ensmp.fr	3,0 ko	10/05/2006 23:42

Message non HTML

```
Auteur  : lerallut
Projet : morphee
Module  : pythonExt

Répertoire     : morphee/pythonExt/src


Modified Files:
        pymImageLib.cpp


Log Message:
Commit et test des ImCreate et ImCreateSame. C'est pour l'infarctus de
Raffi...

http://rhodes.ensmp.fr/cgi-bin/viewcvs.cgi/morphee//morphee/pythonExt/src/pymImageLib.cpp
====================================================================
RCS file: /home/lerallut/cvsroot/morphee/morphee/pythonExt/src/pymImageLib.cpp,v
retrieving revision 1.16
retrieving revision 1.17
diff -u -3 -r1.16 -r1.17
--- pymImageLib.cpp     25 Aug 2005 17:10:42 -0000      1.16
+++ pymImageLib.cpp     10 May 2006 11:04:48 -0000      1.17
@@ -35,6 +35,30 @@
                                "im2 = getSameOf(im, dataCategory, scalarDataType): returns an image of the same size,
but of different types (allocated if im is allocated)");


+
+                       boost::python::def( "ImCreate",
+                                           (ImageInterface* (*)(coord_t,coord_t,coord_t,const std::string&))&ImCreate,
+                                           boost::python::args("xSize","ySize","zSize","sType"),
+                                           boost::python::return_value_policy<boost::python::manage_new_object>(),
+                                           "im = ImCreate( xSize, ySize, zSize, sType ): returns a new image of the
specified type (see C++ documentation for the supported types)");
+
```

AQ: Tests unitaires: *errare humanum est*

- vérifications automatiques et régulières
- interceptent beaucoup de bugs
- actuellement (Octobre 2007) 650 tests automatiques pour le coeur de MorphM (dont 185 dans 'morphoBase')
- L'ensemble des modules d'addons est également testé

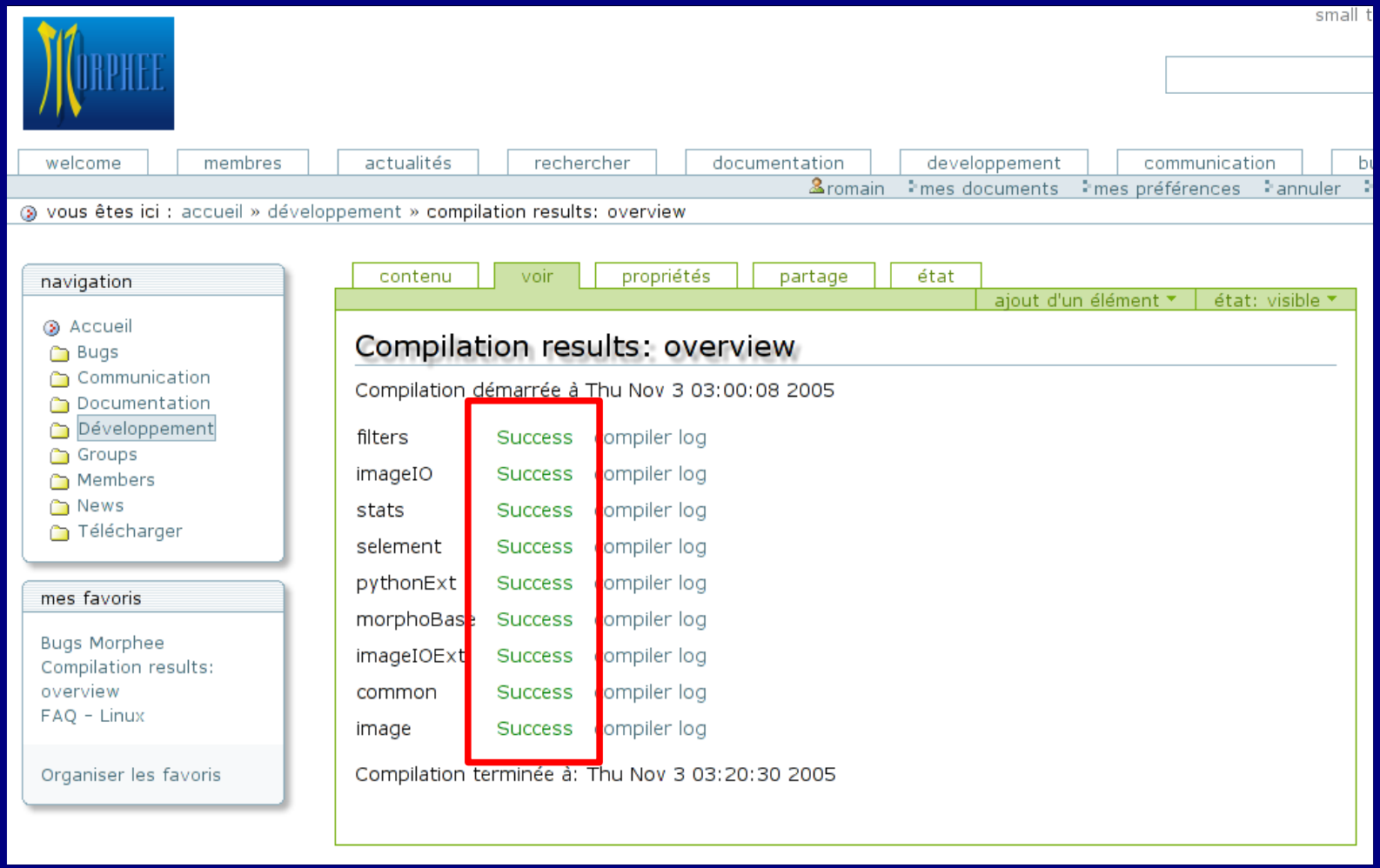
AQ: Tests unitaires: *perseverare diabolicum*

- Éviter les régressions
- Si un bug est découvert:
  - un test déclenchant le bug est ajouté
  - le bug est corrigé
  - les modifications sont ajoutées au CVS
- le test automatique garantit que le bug n'apparaîtra plus

AQ: compilation nocturne

Toutes les nuits (Linux et Windows):

- compilation complète et tests unitaires
- pour le coeur et certains modules externes
- génération de la documentation:



Plan

- Introduction
- Briques
- Interface Python
- Outils collaboratifs et AQ
- Conclusion

Bilan

- l'essentiel des algos de morpho a été porté
- un outil évolutif, de recherche
- taillé pour les besoins du CMM
- utilisé dans plusieurs projets fortement distincts
- accessible sur plusieurs niveaux (C++, Python)
- grande synergie entre les développeurs

Au-delà de la Morphologie

- FFT (covariances, convolutions, ...)
- DirectShow et capture de flux vidéo (Windows)
- E/S tous formats d'images:

  JPEG , JPEG 2000, GIF, BMP, SunRaster, PNG, EPS, TIFF, CSV, ...
- Couleur
  - distances couleur (Lab, HLS, RGB, ...) , gradients
  - espaces couleur et illuminants
- Optimisations avec MM4M, fastmorphm

Qualité professionnelle

- Grande robustesse
- Méthodologie de développement
- Outils de tests et de contrôle de la qualité

Ce qu'il reste à faire

- améliorer la documentation, les tutoriels *(surtout pour ceux qui n'ont pas le source)*
- toolbox Matlab ?
- interface graphique « à la MicroMorph » ?
- publications ?
- **valorisation** ! *(segmentator, etc)*

Vers une implémentation de référence ?