OUTIL LOGICIEL POUR LE TRAITEMENT DES IMAGES

BIBLIOTHÈQUE, PARADIGMES, TYPES ET ALGORITHMES

Thierry Géraud

EPITA-LRDE

25 juin 2012

LE SUJET

Le sujet :

- à l'intersection de l'IMAGE et de l'INFORMATIQUE.

LE SUJET

Le sujet :

- à l'intersection de l'IMAGE et de l'INFORMATIQUE.

Le point de vue du praticien :

- " le traitement d'images est le but, le domaine visé...
- ...et l'informatique est le moyen, l'outil. "

LE SUJET

Le sujet :

- à l'intersection de l'IMAGE et de l'INFORMATIQUE.

Le point de vue du praticien :

- " le traitement d'images est le but, le domaine visé...
- ...et l'informatique est le moyen, l'outil. "

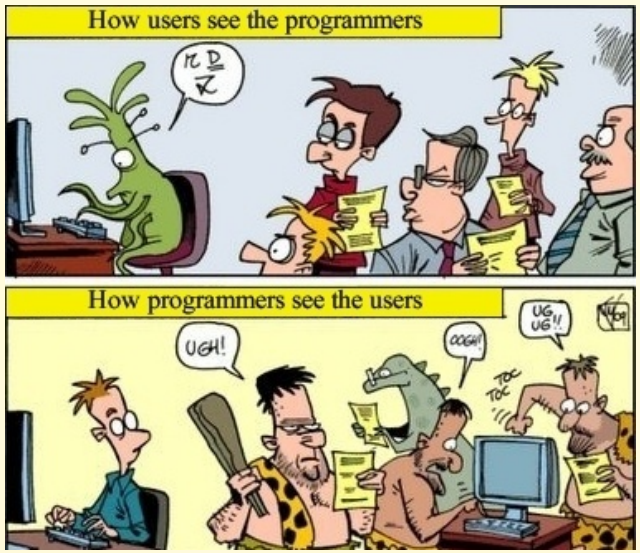
Ici, l'outil est notre but !

une bibliothèque...

Le sujet

Deux mondes différents :

- pas le même langage,
- pas les mêmes connaissances,
- pas les mêmes attitudes / réflexes.



Le sujet

Deux mondes différents :

- pas le même langage,
- pas les mêmes connaissances,
- pas les mêmes attitudes / réflexes.

Le paradoxe du traiteur d'images :

- il dit "ne pas faire de l'informatique"
  il est utilisateur de l'outil (les fonctionnalités d'une bibliothèque)
- pourtant il dit "mon programme" et "je débogue"
  il est producteur de code !

LE MÉTA-MESSAGE

Transférer des connaissances et offrir un outil puissant.

En filigrane : est-ce que son outil l'aide beaucoup à bien produire du bon code ?

DES IDÉES REÇUES ET CONTRE-VÉRITÉS

Les classiques :

- C'est "normal" de manipuler des pointeurs.
- Matlab® est amplement suffisant.
- On est souvent obligé de copier-coller-modifier.
- ...

DES IDÉES REÇUES ET CONTRE-VÉRITÉS

À propos de *généricité* :

- Avec une bibliothèque générique, on est obligé de “coder générique”.
- Si c’est générique, c’est beaucoup plus difficile pour l’utilisateur...
- ...ça nécessite pas mal de connaissances en informatique.

- Une bibliothèque est générique ou ne l’est pas.
- Pour être générique, il suffit de rajouter le bon mot-clef.

Des idées reçues et contre-vérités

Pire :

- si l'on a besoin de traiter
  - des images 3D, des images énormes, des vidéos, des graphes,
  - ...
- il est "normal"
  - que l'outil ne puisse pas le faire
  - qu'il faille un outil *ad hoc*.

$\longrightarrow$ L'outil a un problème de limitation.

acceptation du périmètre v. changement d'outil

UNE ILLUSTRATION EN 2D

Deux algorithmes très classiques :

entrée

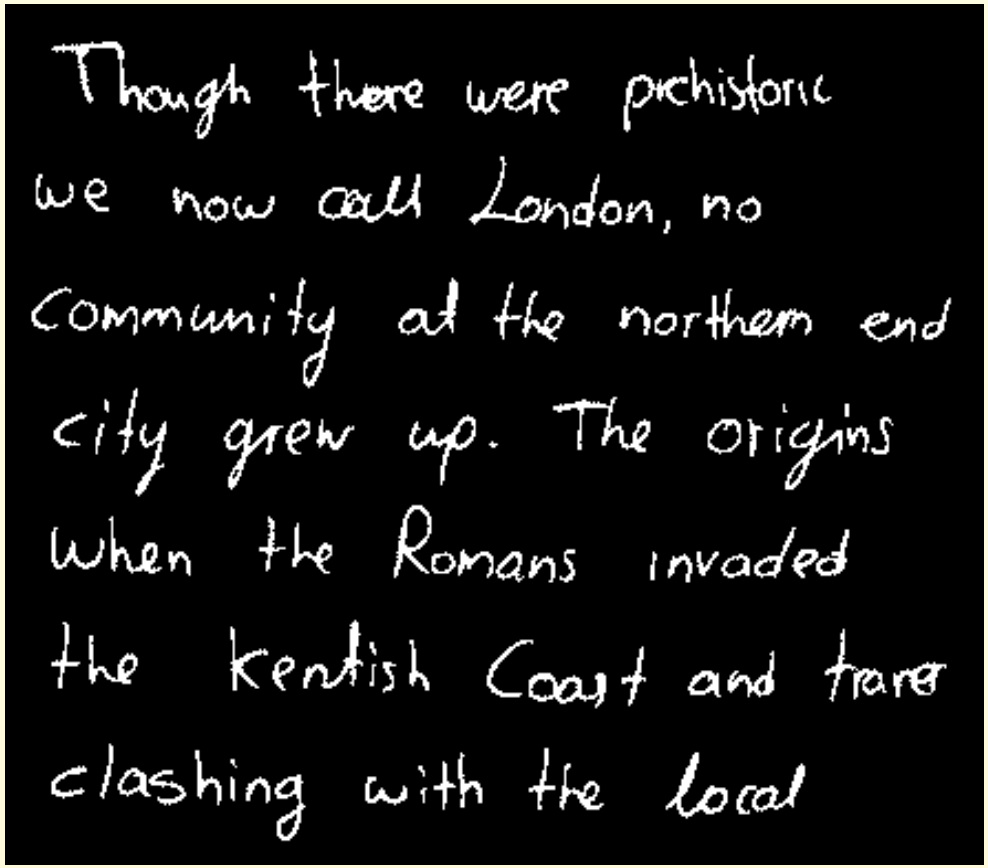
étiquetage

zones d'influence

...fournis par toute bibliothèque.

UNE ILLUSTRATION EN 2D

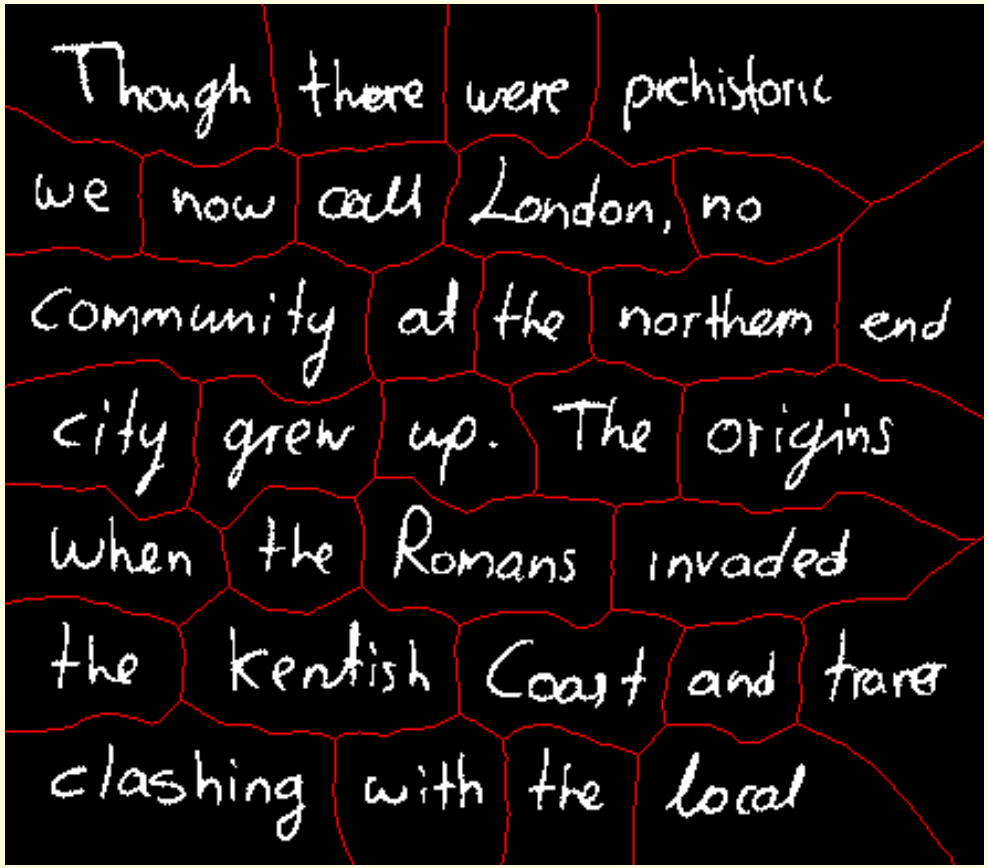
Le problème à résoudre : identifier les lignes de texte...



entrée

UNE ILLUSTRATION EN 2D

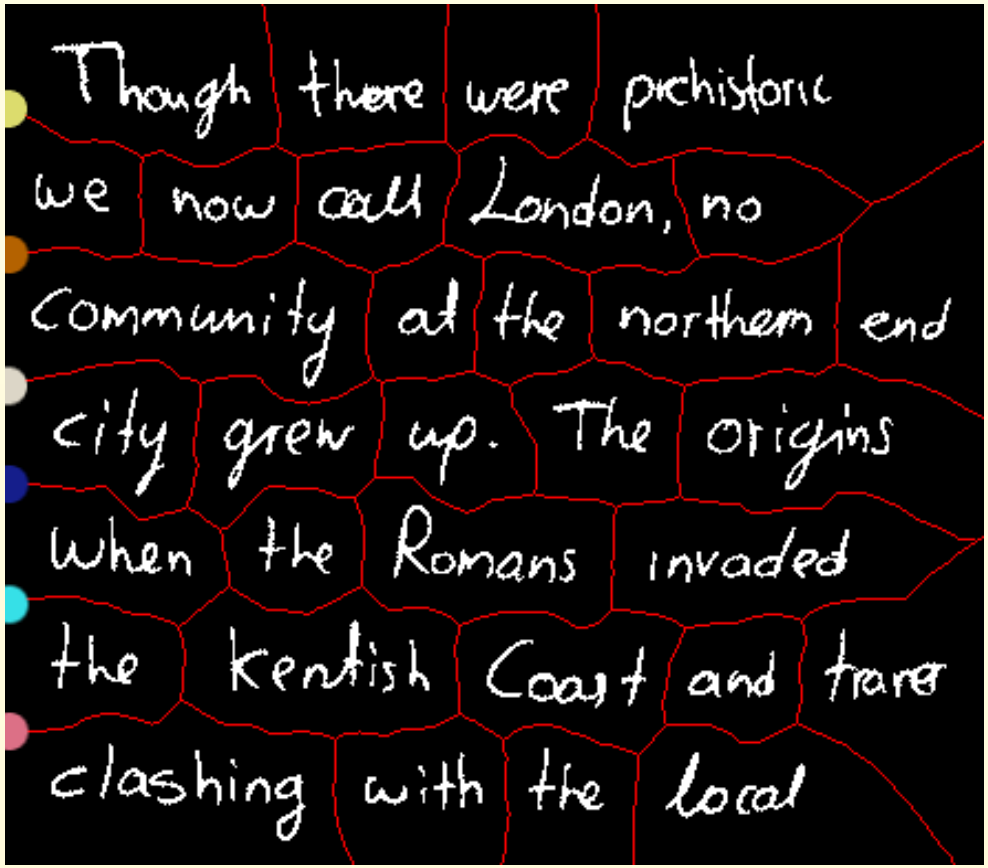
On dispose de ce résultat intermédiaire :



segmentation en mots

UNE ILLUSTRATION EN 2D

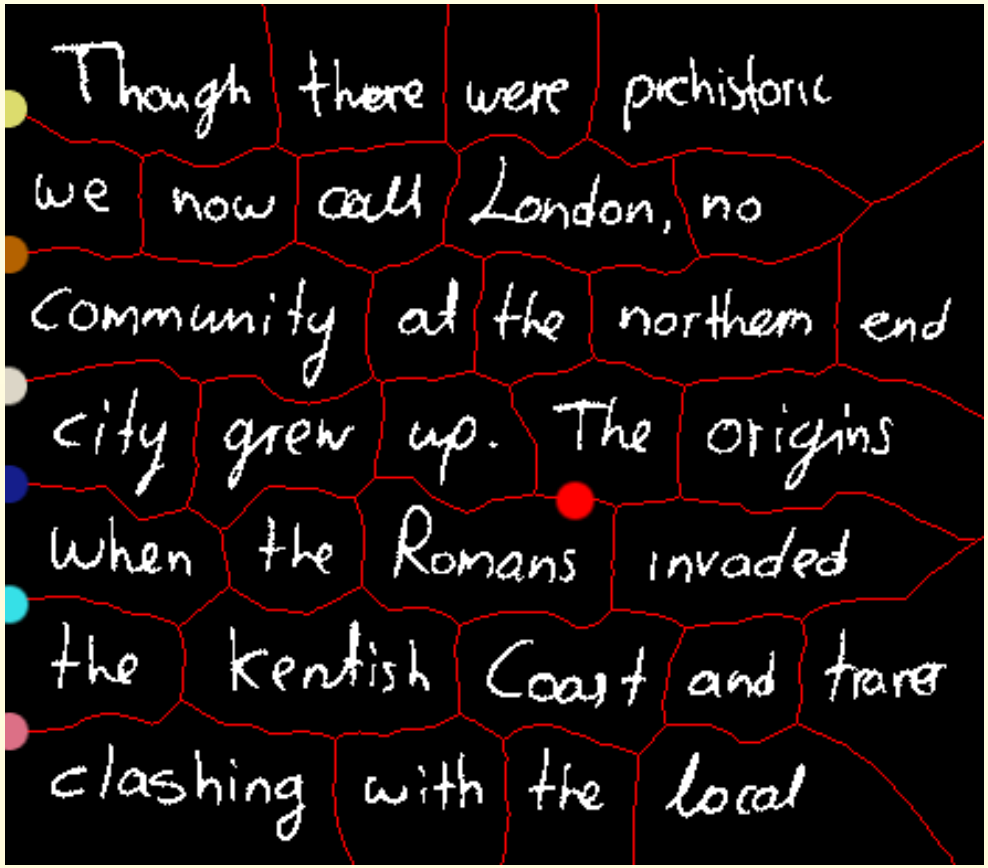
Une idée :



identifier les interlignes = étiqueter les points de la première colonne

UNE ILLUSTRATION EN 2D

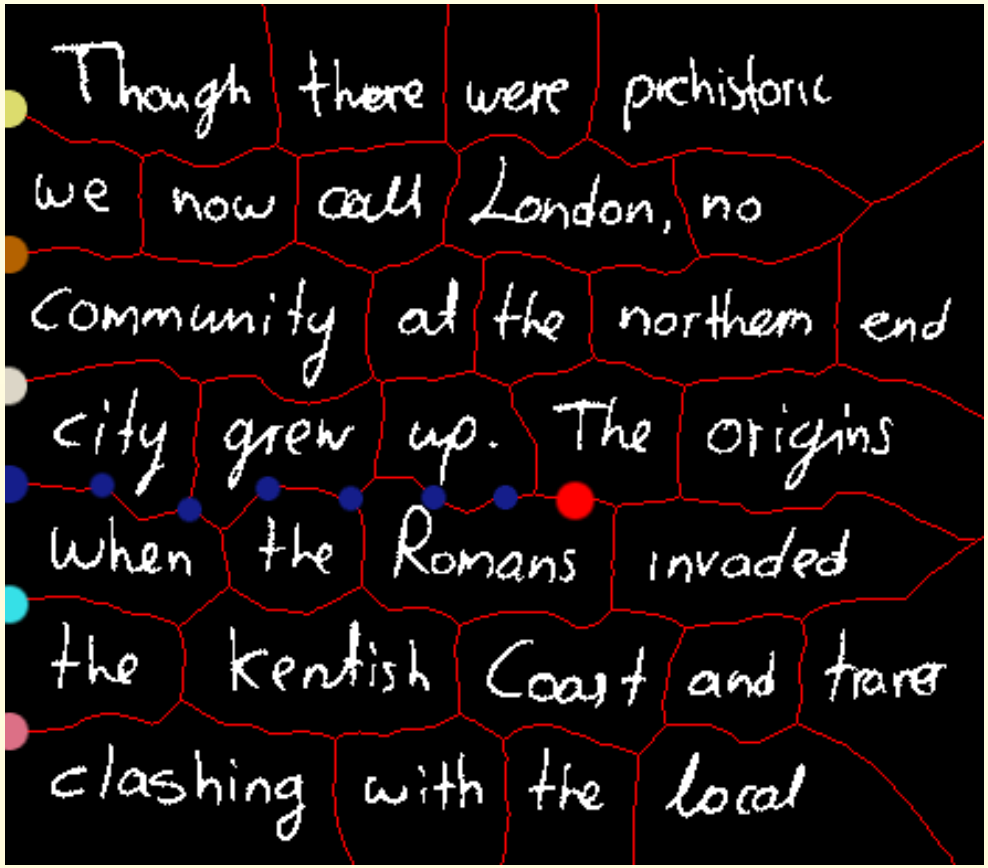
Suite de l'idée :



un point appartient à un interligne...

UNE ILLUSTRATION EN 2D

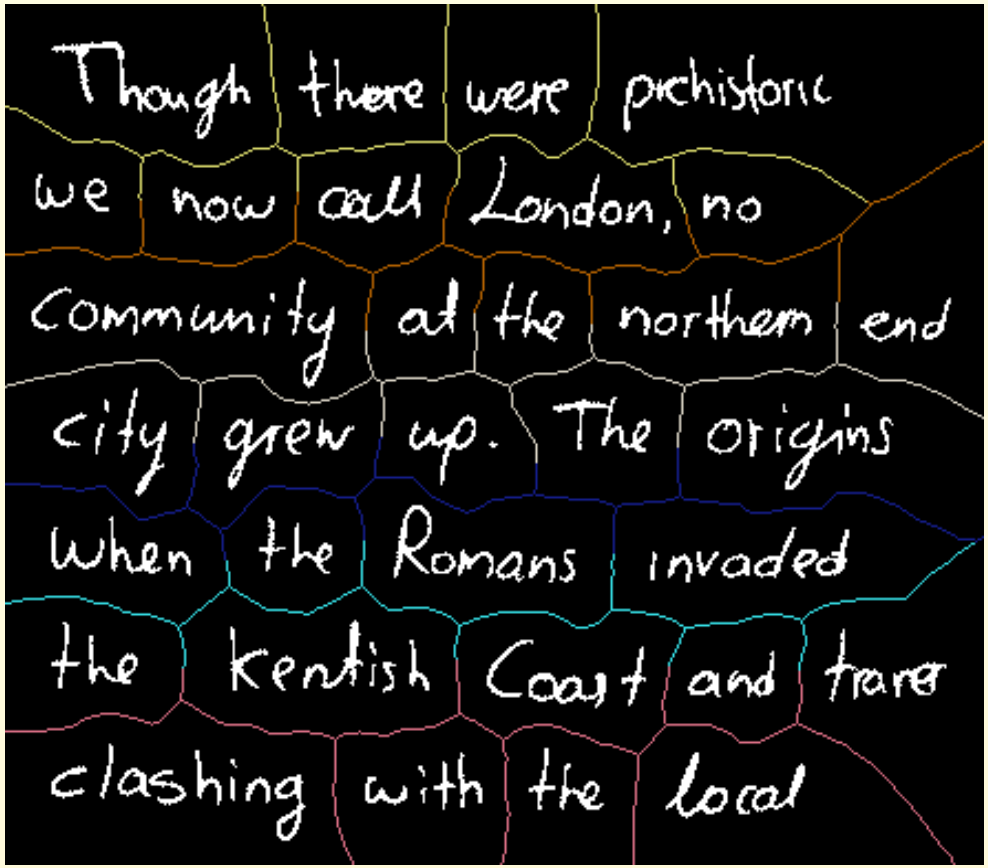
Suite de l'idée :



...s'il est dans la zone d'influence de l'étiquette correspondante

UNE ILLUSTRATION EN 2D

On s'attend à obtenir facilement...



...ce résultat

UNE ILLUSTRATION EN 2D

Mais :

- dans les faits, le praticien sera bloqué, $\longrightarrow$ algorithme non ré-utilisable
- il lui faudra écrire du code, $\longrightarrow$ informatique v. métier
- en cela, l'outil est limitatif.

MOTIVATION DE LA DÉMARCHE

Les motivations :

- non-satisfaisait par l'état de l'art des bibliothèques,
- persuadé qu'on peut faire mieux du point de vue informatique, contributions
- sujet à couleur informatique sur un domaine que je connais,
- pas seulement "théorique" mais aussi applicatif,
- sujet "orienté-projet". sous-sujets / plusieurs domaines / logiciel

MOTIVATION DE LA DÉMARCHE

Les motivations :

- non-satisfaisait par l'état de l'art des bibliothèques,
- persuadé qu'on peut faire mieux du point de vue informatique, contributions
- sujet à couleur informatique sur un domaine que je connais,
- pas seulement "théorique" mais aussi applicatif,
- sujet "orienté-projet". sous-sujets / plusieurs domaines / logiciel

MOTIVATION DE LA DÉMARCHE

Les motivations :

- non-satisfaisait par l'état de l'art des bibliothèques,
- persuadé qu'on peut faire mieux du point de vue informatique, contributions
- sujet à couleur informatique sur un domaine que je connais,
- pas seulement "théorique" mais aussi applicatif,
- sujet "orienté-projet". sous-sujets / plusieurs domaines / logiciel

MOTIVATION DE LA DÉMARCHE

Les motivations :

- non-satisfaisait par l'état de l'art des bibliothèques,
- persuadé qu'on peut faire mieux du point de vue informatique, contributions
- sujet à couleur informatique sur un domaine que je connais,
- pas seulement "théorique" mais aussi applicatif,
- sujet "orienté-projet". sous-sujets / plusieurs domaines / logiciel

MOTIVATION DE LA DÉMARCHE

Les motivations :

- non-satisfaisait par l'état de l'art des bibliothèques,
- persuadé qu'on peut faire mieux du point de vue informatique, contributions
- sujet à couleur informatique sur un domaine que je connais,
- pas seulement "théorique" mais aussi applicatif,
- sujet "orienté-projet". sous-sujets / plusieurs domaines / logiciel

Le problème

- Le quoi :
  traduire les entités du domaine en entités logicielles. ⟶ bibliothèque
- L'objectif principal :
  ne pas limiter l'utilisateur. ⟶ haut degré de généricité
- Le comment :
  obtenir un paradigme qui garantit les objectifs. ⟶ plus que programmer

LE PROBLÈME

- Le quoi :
  traduire les entités du domaine en entités logicielles. ⟶ bibliothèque
- L'objectif principal :
  ne pas limiter l'utilisateur. ⟶ haut degré de généricité
- Le comment :
  obtenir un paradigme qui garantit les objectifs. ⟶ plus que programmer

LE PROBLÈME

- Le quoi :
  traduire les entités du domaine en entités logicielles. ⟶ bibliothèque
- L'objectif principal :
  ne pas limiter l'utilisateur. ⟶ haut degré de généricité
- Le comment :
  obtenir un paradigme qui garantit les objectifs. ⟶ plus que programmer

L'OBJECTIF

LA COULEUR DE LA BIBLIOTHÈQUE :

Privilégier le potentiel de fonctionnalités (*capability*)
par rapport aux fonctionnalités disponibles (*availability*).

Donc :

- une bibliothèque qui n'est pas dédiée
  - à un sous-domaine applicatif,
  - à une certaine classe de traitements,
- une bibliothèque où "tout" est ou sera possible.

Les contraintes

En corollaire :

- Une bibliothèque “vraiment” utilisable :
  - simplicité,
  - performances.
- Une bibliothèque dont le potentiel est efficient :
  - des additions à moindre coût,
  - des additions qui décuplent les fonctionnalités.

Comment ? Trouver le paradigme adéquat...

Choix d'un paradigme

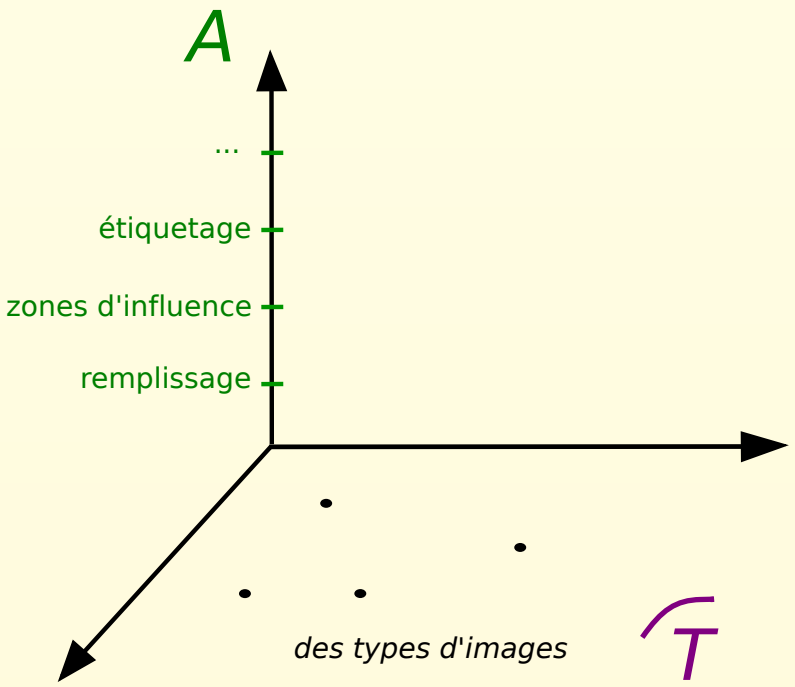
Paradigmes	Sûreté du typage	Complexité du code	Efficacité	Abstractions explicites	Une impl. par algorithm
Duplication de code	✓	✗	✓	✗	✗
Généralisation	✗	=	=	✗	✓
Programmation Orientée-Objet (OOP)	=	✓	✗	✓	✓
Programmation Générique (GP)	✓	✓	✓	=	✓

Choix du monde statique (GP).

$\longrightarrow$ validité + performances + covariance

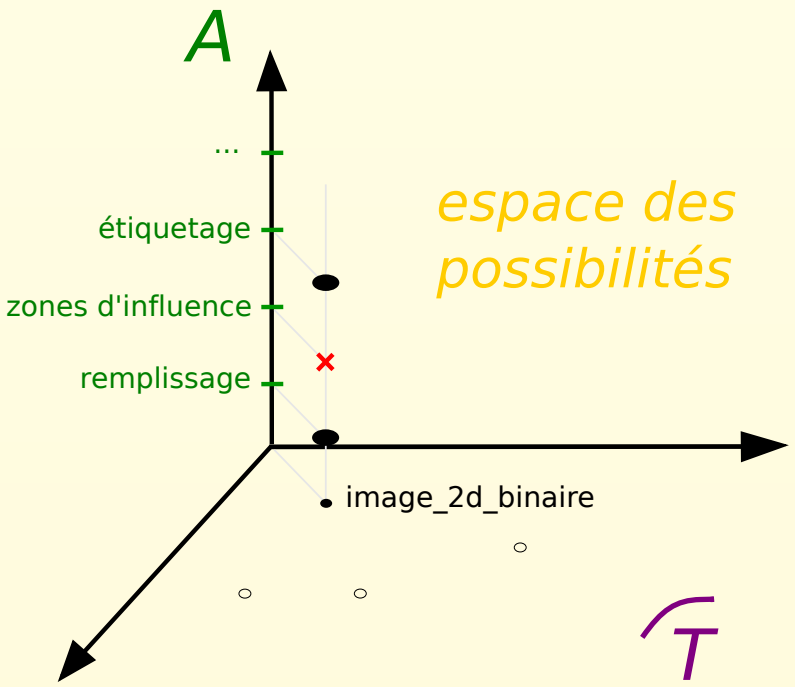
$\longrightarrow$ non-limitatif + efficient

La généricité



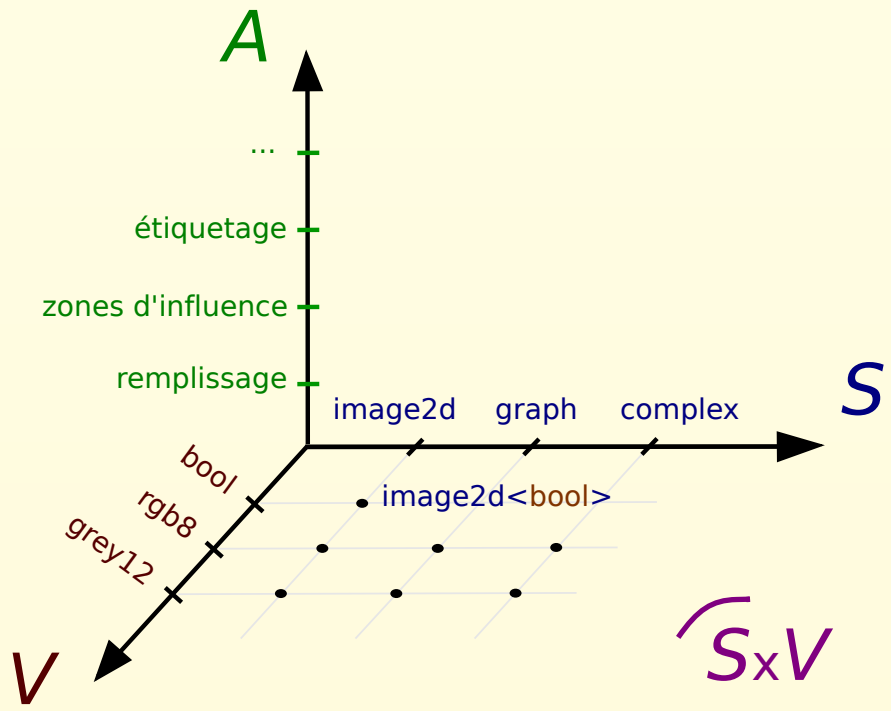
une représentation classique “algorithmes v. types de données”

La généricité



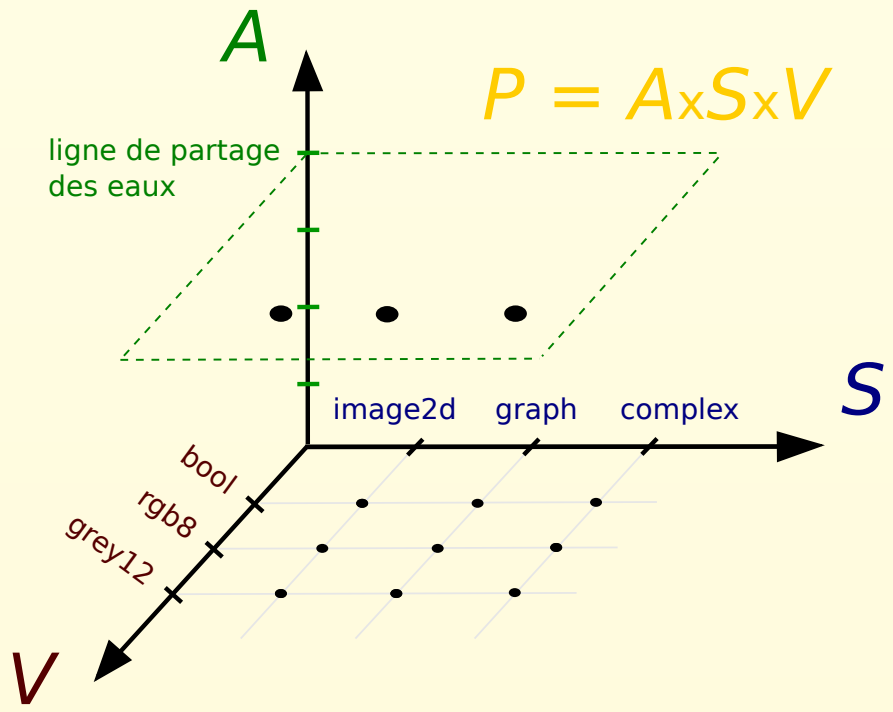
ne pas être limité = couvrir au maximum l'espace des possibles

La généricité



découplage : $S + V$ implémentations seulement

La généricité



\+ une unique implémentation par algorithme

La généricité

La programmation générique

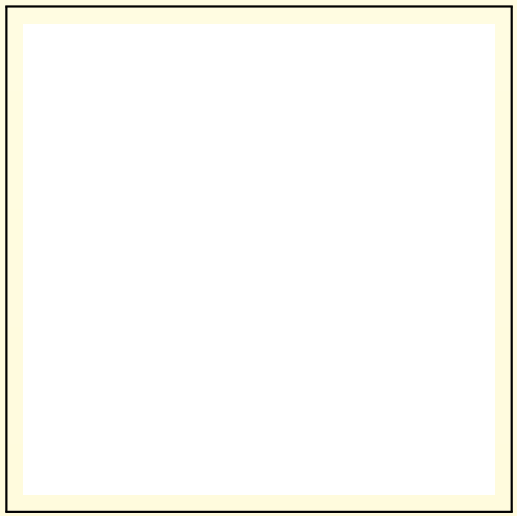
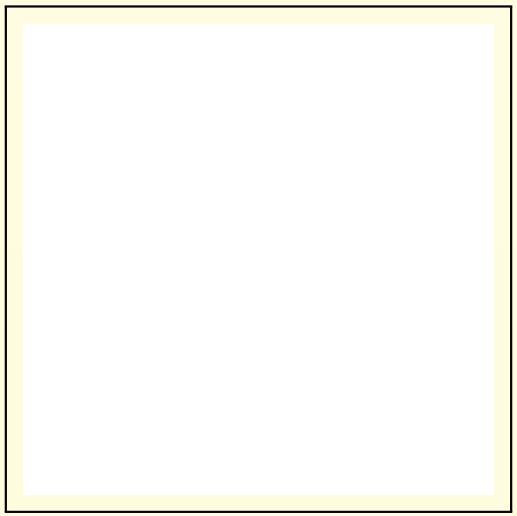
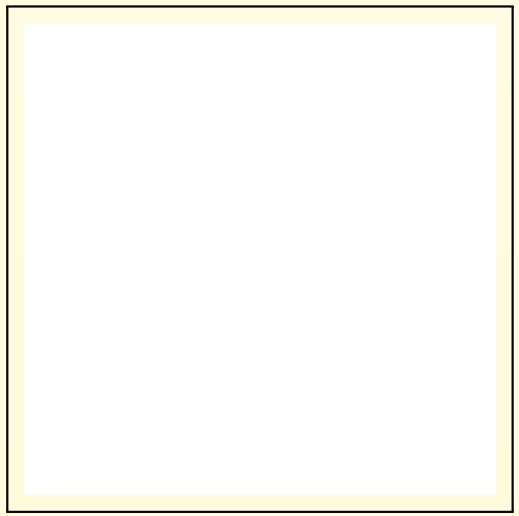
est un outil de base

- pour découpler une entité et les types qu'elle manipule,
- pour rendre efficiente toute addition de fonctionnalités.

En pratique, algorithmes non abstraits et combinatoire pas atteinte...

La généricité

Quels types veut-on avoir ?

Quelle bibliothèque en est là ?

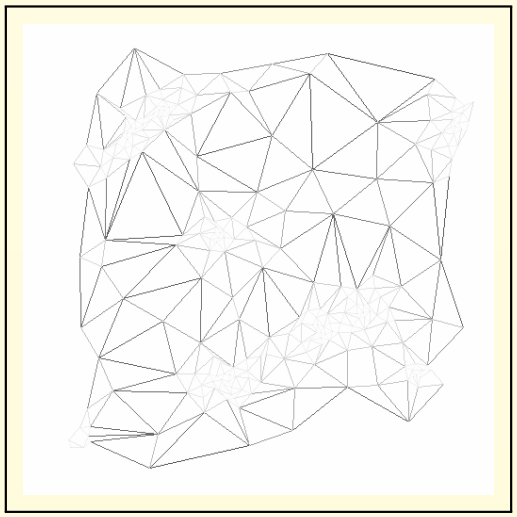
La généricité

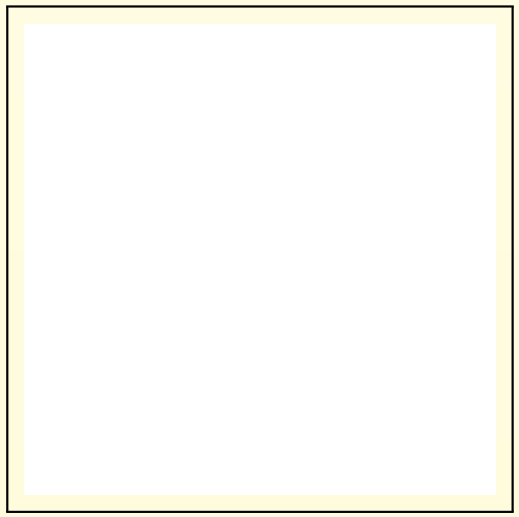
Quels types veut-on avoir ?

Quelle bibliothèque en est là ?

La généricité

Quels types veut-on avoir ?

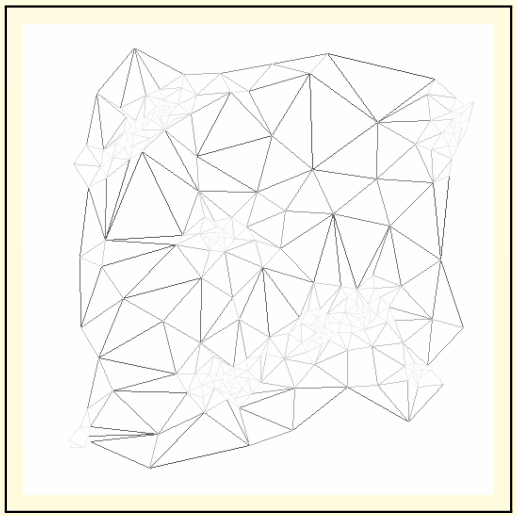




Quelle bibliothèque en est là ?

La généricité

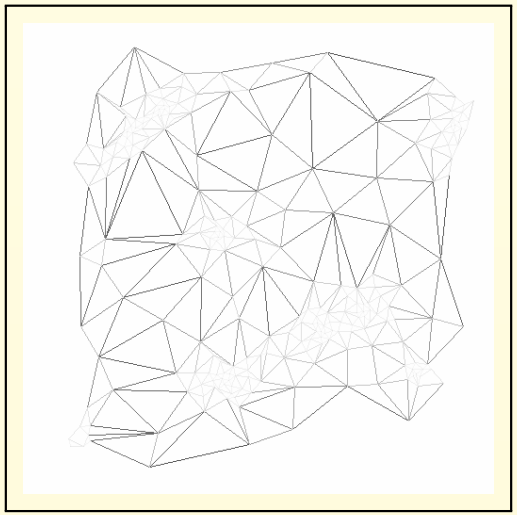
Quels types veut-on avoir ?



Quelle bibliothèque en est là ?

La généricité

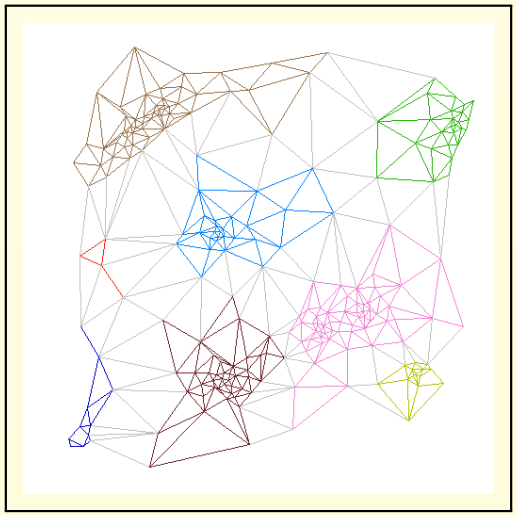
Quels types peut-on traiter ?



Quelle bibliothèque en est là ?

La généricité

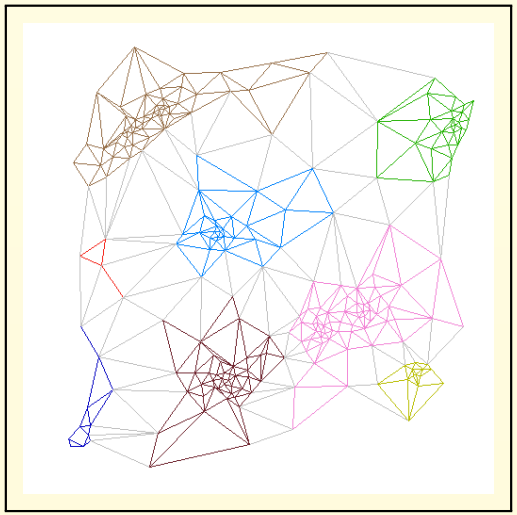
Quels types peut-on traiter ?

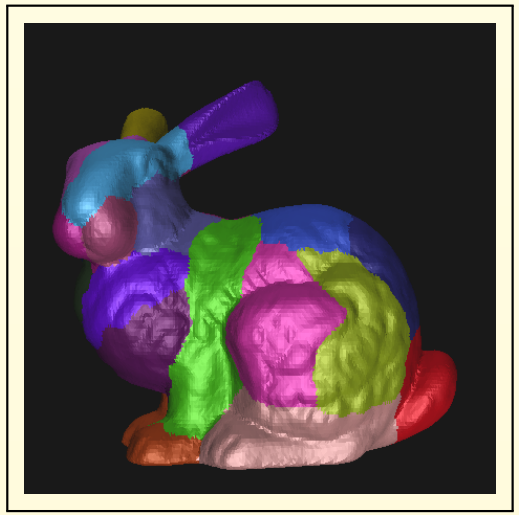


Quelle bibliothèque en est là ?

La généricité

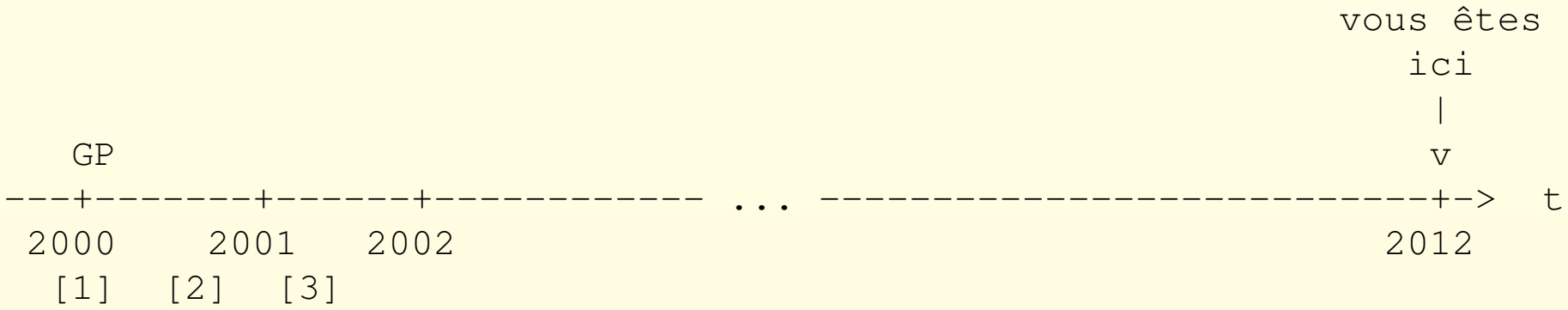
Quels types peut-on traiter ?





Quelle bibliothèque en est là ?

PARADIGMES ET TYPES



[1] Programmation générique (GP) :

Obtaining Genericity for Image Processing and Pattern Recognition Algorithms. ICPR, 2000.

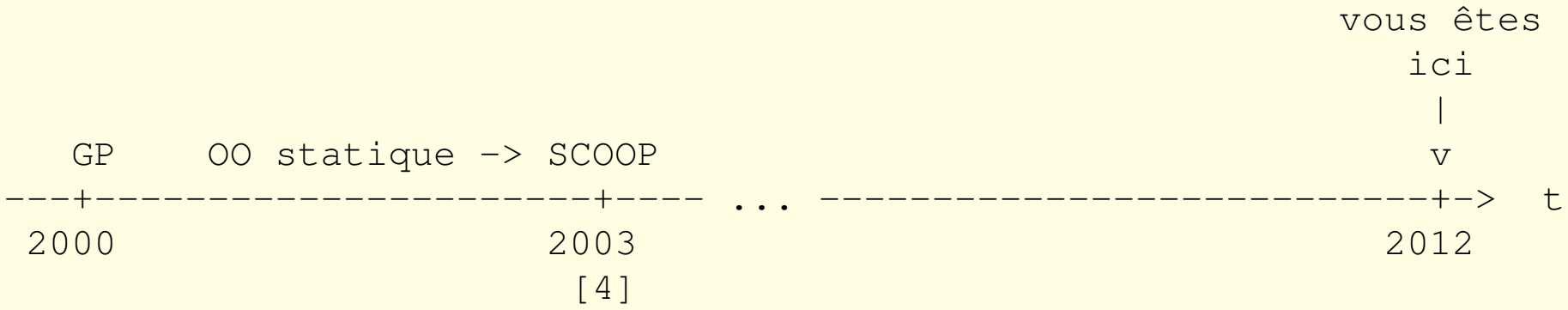
[2] Compatibilité avec le monde dynamique :

A Component-Based Platform Mixing Generic, Generative & OO Programming. Young Researchers Workshop @ GCSE, 2000.

[3] Formalisation des patrons de conception statiques :

Generic Programming Redesign of Patterns. EuroPLoP, 2001.

PARADIGMES ET TYPES

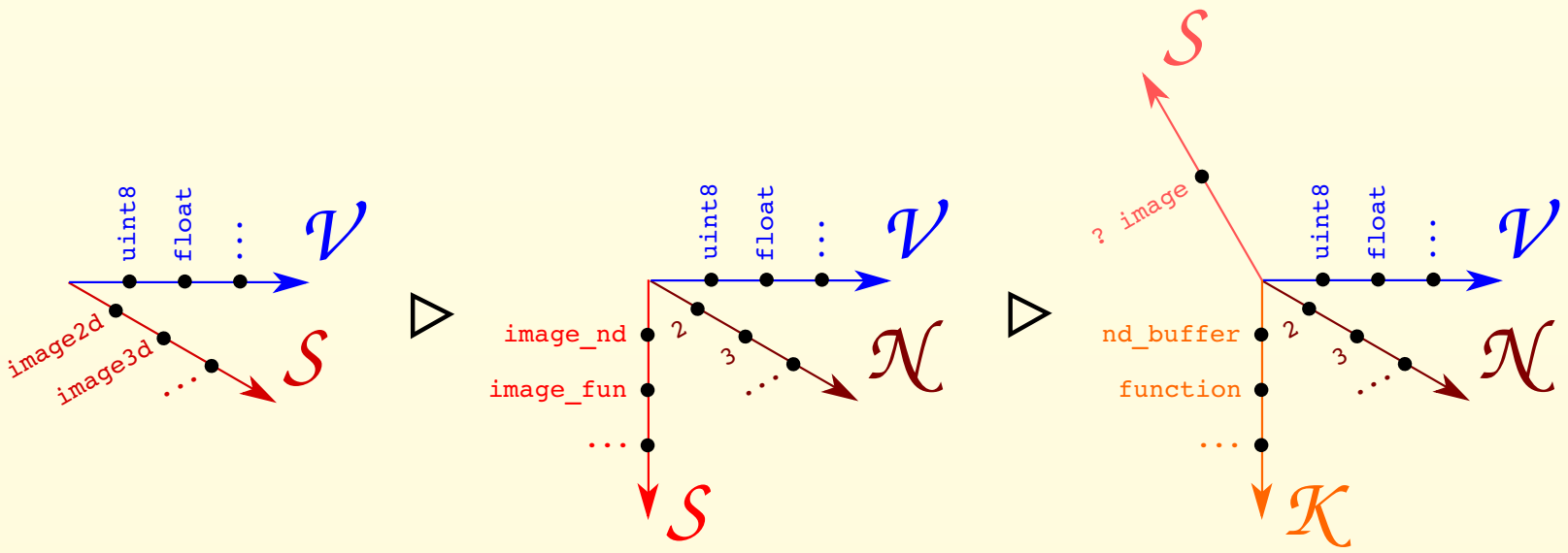


Paradigmes	Sûreté du typage	Complexité du code	Efficacité	Abstractions explicites	Une impl. par algorithm
Programmation Orientée-Objet (OOP)	=	✓	✗	✓	✓
Programmation Générique (GP)	✓	✓	✓	=	✓

[4] Mélange entre programmation orienté-objet et programmation générique :
A Static C++ Object-Oriented Programming. Workshop MPOOL @ OOPSLA, 2003.

Le hic

La vision des types en deux axes est simpliste, et ses raffinements également :



Les problèmes :

- les axes sont inévitablement couplés, $\longrightarrow$ paramètres intriqués
- le choix de ces axes est limitatif en soi ! $\longrightarrow$ géodésie du tore ?

Le hic

En fait :

- l'hyperplan des types se définit par un ensemble de propriétés,
- un type particulier (simplissime) est un point de cet espace.

Entre-aperçu de certaines propriétés :

- peut-on modifier les valeurs des pixels ?
- les valeurs sont-elles faiblement quantifiées ?
- les sites sont-ils sur une grille ?
- est-ce que le nombre de sites est connu ?
- a-t-on une association "1 site / 1 valeur" ?
- peut-on jouer avec l'arithmétique des pointeurs ?
- etc.

Exemple : `image2d<T>`...

UN MULTI-PARADIGME POUR LE CALCUL SCIENTIFIQUE :

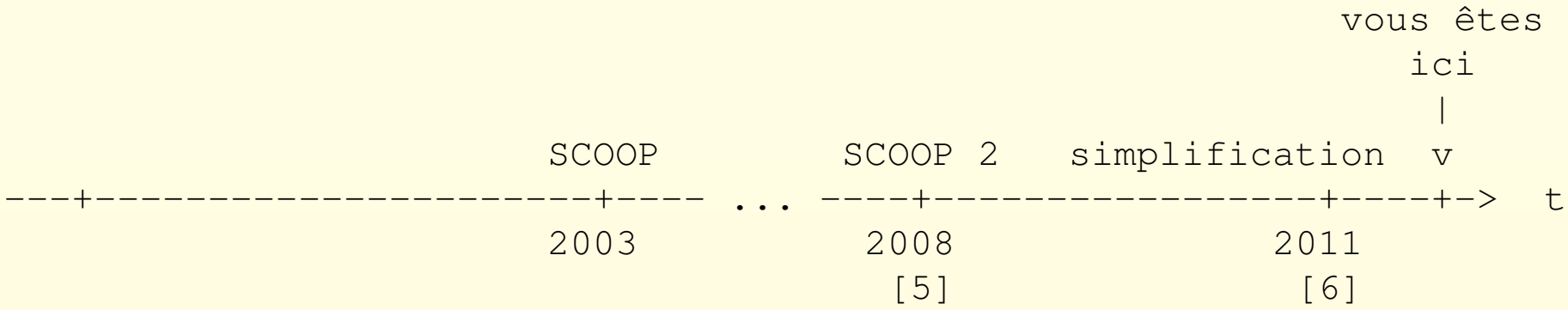
Principe :

- propriétés d'un type données sous forme déclarative,
- abstractions matérialisées sous forme de classes,
- généricité et typage statique.

Quelques caractéristiques :

- types virtuels,
- conteneurs covariants,
- polymorphisme paramétrique borné,
- multi-méthodes,
- des "classes génériques à interface polymorphe"...

PARADIGMES ET TYPES



[5] Programmation par propriétés :

Semantics-Driven Genericity: A Sequel to SCOOP. Workshop MPOOL @ OOPSLA, 2008.

[6] Simplification du paradigme :

Towards a Software Architecture for Generic Image Processing. Roland Levillain PhD report, 2011.

L'IMPACT

LA QUESTION :

Qu'est-ce qu'une image ?

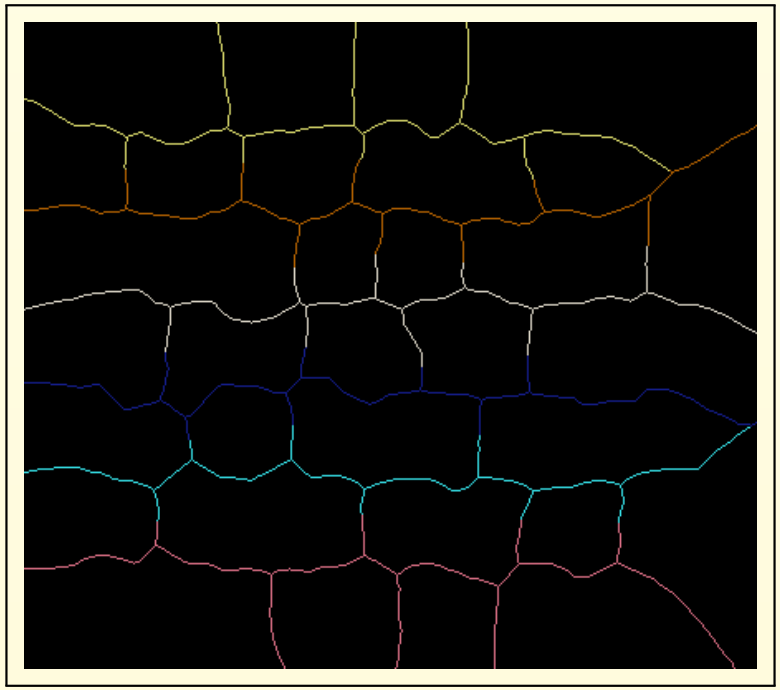
LA RÉPONSE :

Une interface générale, commune à toutes les images,
+ des interfaces spécifiques en fonction des propriétés.

$\rightarrow$ cela conduit à une taxonomie par propriétés...

originalité v. taxonomie ϵ ailleurs + propriétés explicites

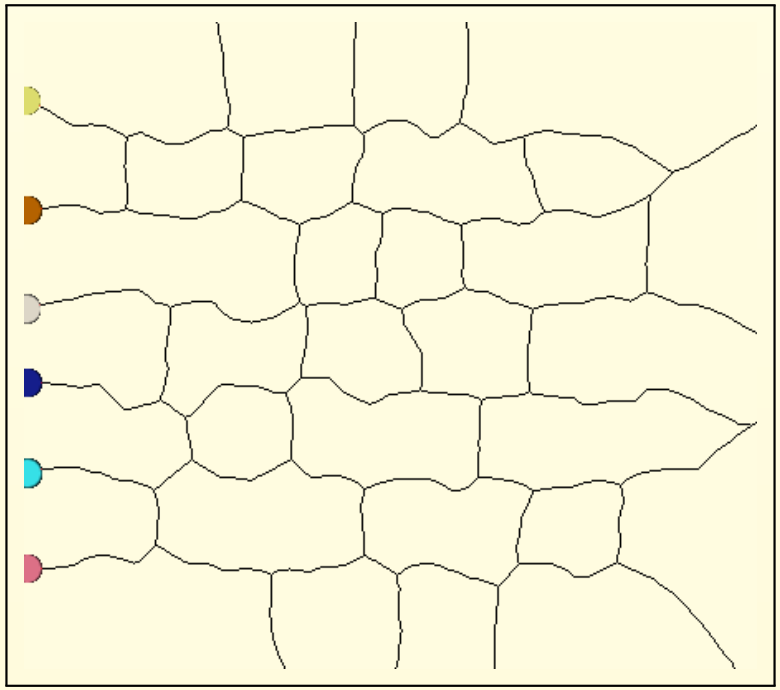
REPRISE DE L'EXEMPLE



sortie

Quelle image doit être en entrée du calcul d'IZ ?

REPRISE DE L'EXEMPLE



entrée

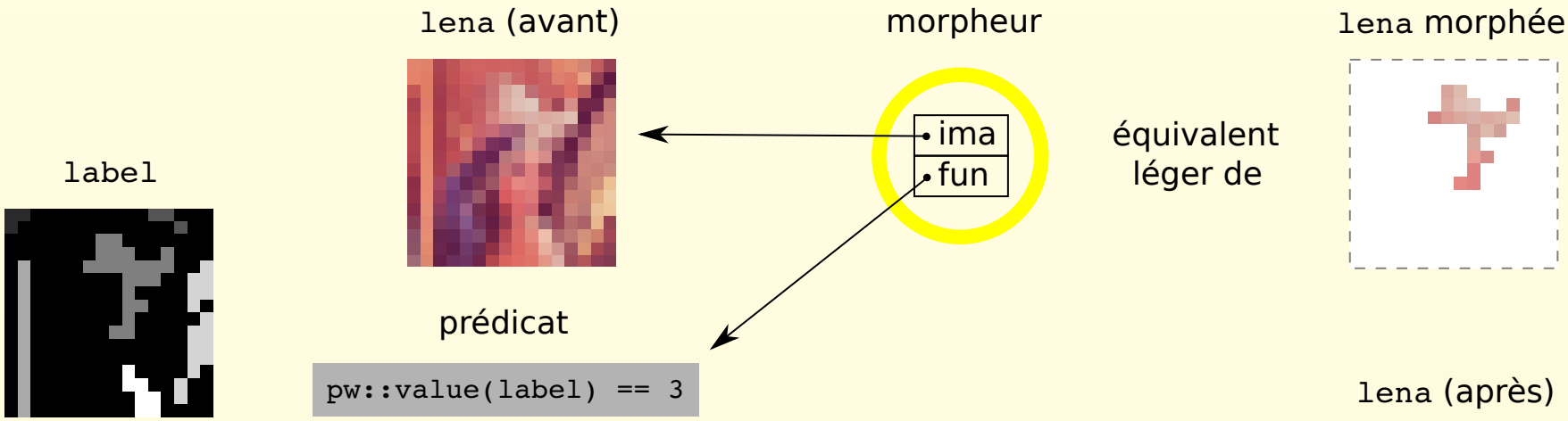
→ quel est ce type d'images ?

Les morpheurs

Les *morpheurs* : des objets qui “modifient” des objets existants.

→ comme certains *design patterns*

Les morpheurs



appel à un algorithme :

```
data::fill((lena | (pw::value(label) == 3)).rw(), literal::green)
```

Les morpheurs

Différentes catégories :

- modification
  - du domaine,
  - des valeurs,
  - des deux à la fois,
- addition de comportement.

Leur définition est générique :

```
image_if<I,F>  operator | (Image<I>& ima, const Function_v2b<F>& fun)
```

Un modèle d'efficience :

- nouveaux axes qui s'appliquent à l'hyperplan des types,
- modification extrusive des algorithmes.

Les types

Une estimation :

- plus de 100 types différents utilisés ces dernières années,
- v. généralement moins de 10 types pour les autres bibliothèques.

LE MÉTA-MESSAGE

Le méta-message :

Transférer des connaissances et offrir un outil puissant.

En filigrane :

est-ce que son outil l'aide beaucoup à écrire du bon code ?

→ des contributions de différentes natures sur l'écriture de traitements

La bibliothèque et les clients

Quatre catégories de “clients” :

- les assembliers (de fonctionnalités)
- les créateurs (d’algorithmes)
- les fournisseurs (de types)
- les architectes (du cœur)

↑ temps×humain passé dans chaque catégorie

↓ niveau de difficulté

La bibliothèque et les clients

On veut modifier les usages :

catégorie	*%age classique*	*%age idéal*
assemblier	60	> 90
créateur	40	< 10
fournisseur	10	0^+
architecte	ϵ	ϵ

praticien $\approx$ assemblier qui parle son langage

LES TRAITEMENTS

Un quasi langage spécifique au domaine :

$\mathcal{D}$: domaine de *ima*

$\mathcal{N}$: voisinage

V : type de valeur de *ima*

$\forall p \in \mathcal{D}$

$out(p) \leftarrow \inf(V)$

$\forall n \in \mathcal{N}(p)$

$out(p) \leftarrow \sup(out(p), ima(n))$

mln_piter(I) $p(\mathcal{D})$; // $p \in \mathcal{D}$

mln_niter(N) $n(\mathcal{N}, p)$; // $n \in \mathcal{N}(p)$

typedef mln_value(I) V;

for_all(p)

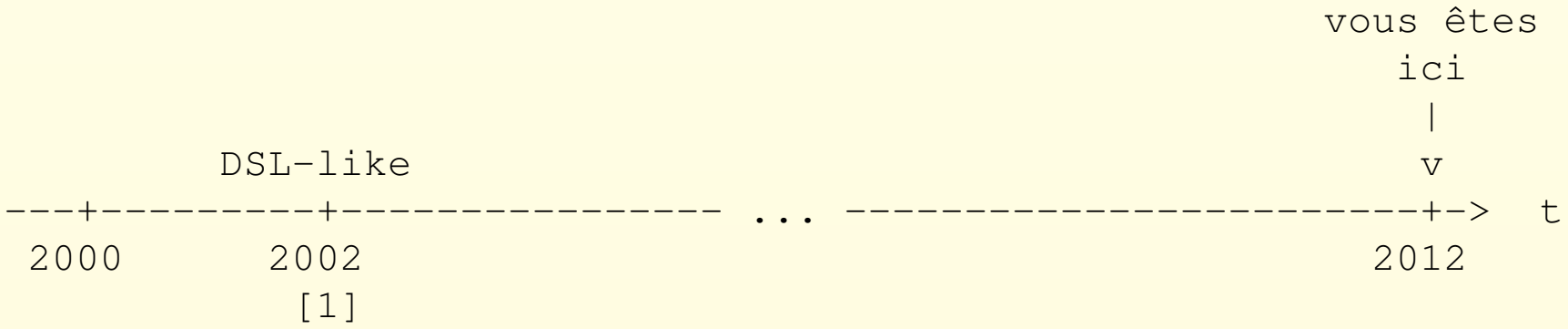
$out(p)$ = mln_inf(V);

for_all(n) if (ima . has(n))

$out(p)$ = $\sup(out(p), ima(n))$;

$\longrightarrow$ pas de détail d'implémentation limitatif

LES TRAITEMENTS

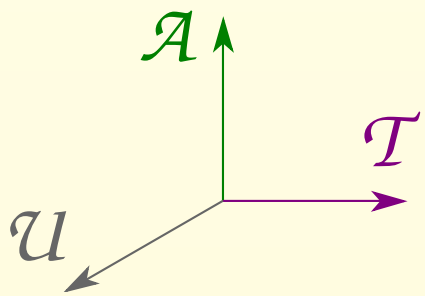


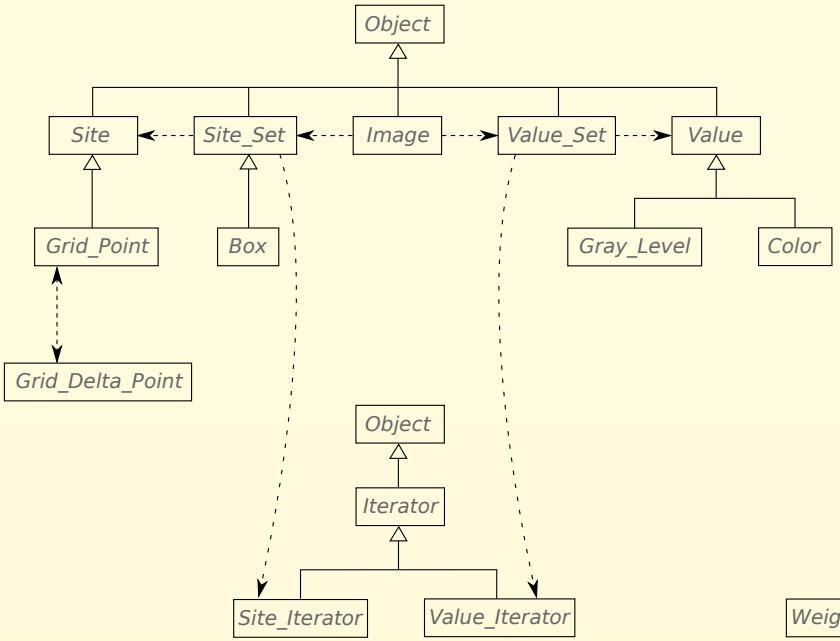
[1] Parler le langage du domaine :

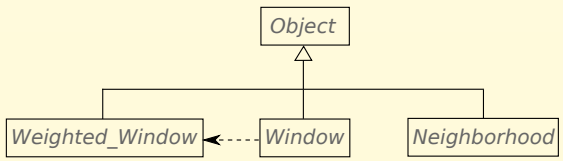
Generic Implementation of Morphological Image Operators. ISMM, 2002.

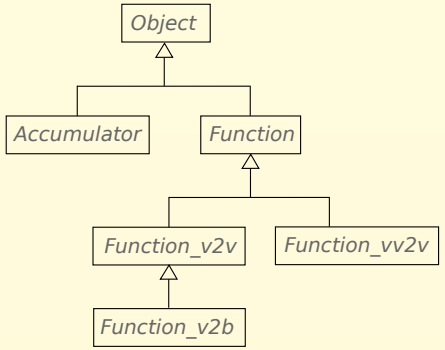
⟶ c'est possible grâce aux outils...

on a développé l'axe des outils :



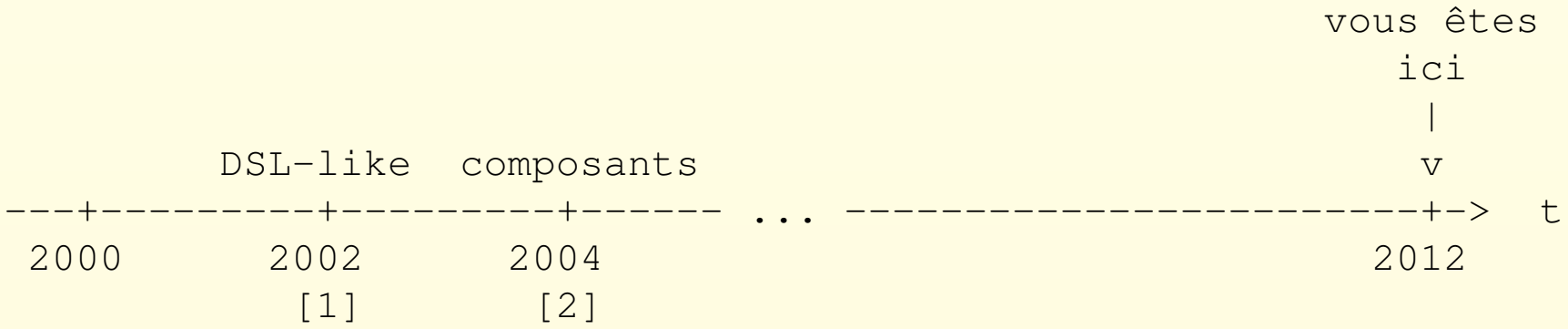






v. autres libs, glu, paradigme et DSL

Les traitements



[1] Parler le langage du domaine :

Generic Implementation of Morphological Image Operators. ISMM, 2002.

[2] Approche composants :

Generic Algorithmic Blocks Dedicated to Image Processing. PhD Workshop @ ECOOP, 2004.

$\longrightarrow$ un algorithme est un composant !

Vers un méta-algorithme...

$\mathcal{D}$: domaine de *ima*

$\mathcal{N}$: voisinage

V : type de valeur de *ima*

$\forall p \in \mathcal{D}$ ...

$out(p) \leftarrow \inf(V)$ // initialisation

$\forall n \in \mathcal{N}(p)$

$out(p) \leftarrow \sup(out(p), ima(n))$ // ?

Vers un méta-algorithme...

$\mathcal{D}$: domaine de *ima*

$\mathcal{N}$: voisinage

oper : opérateur sur un ensemble

$\forall p \in \mathcal{D}$

    oper . init()

    $\forall n \in \mathcal{N}(p)$

        oper . take($ima(n)$)

    $out(p) \leftarrow$ oper . result()

...

Vers un méta-algorithme...

$\mathcal{D}$: domaine de *ima*

$\mathcal{N}$: voisinage

oper : opérateur sur un ensemble

$\forall\, p \in \mathcal{D}$

  oper . init()

  $\forall\, n \in \mathcal{N}(p)$

    oper . take($ima(n)$)

  $out(p) \leftarrow$ oper . result()

mln_piter(I) $p(\mathcal{D})$; // $p \in \mathcal{D}$

mln_niter(N) $n(\mathcal{N}, p)$; // $n \in \mathcal{N}(p)$

// oper est passé en argument du méta-algorithme

for_all(p)

  oper . init()

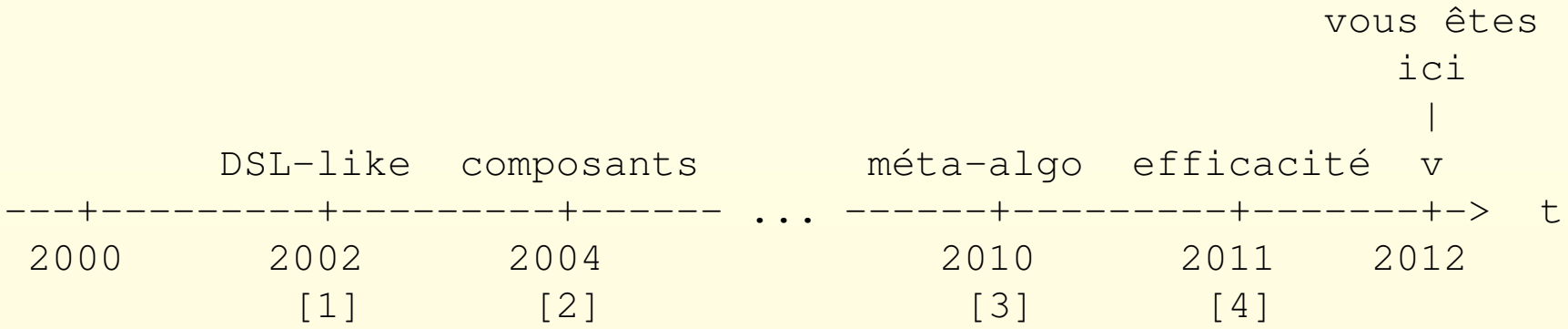
  for_all(n) if (ima . has(n))

    oper . take($ima(n)$)

  $out(p) \leftarrow$ oper . result()

de nombreux avantages...

Les traitements



[1] Parler le langage du domaine :

Generic Implementation of Morphological Image Operators. ISMM, 2002.

[2] Approche composants :

Generic Algorithmic Blocks Dedicated to Image Processing. PhD Workshop @ ECOOP, 2004.

[3] Méta-algorithmes :

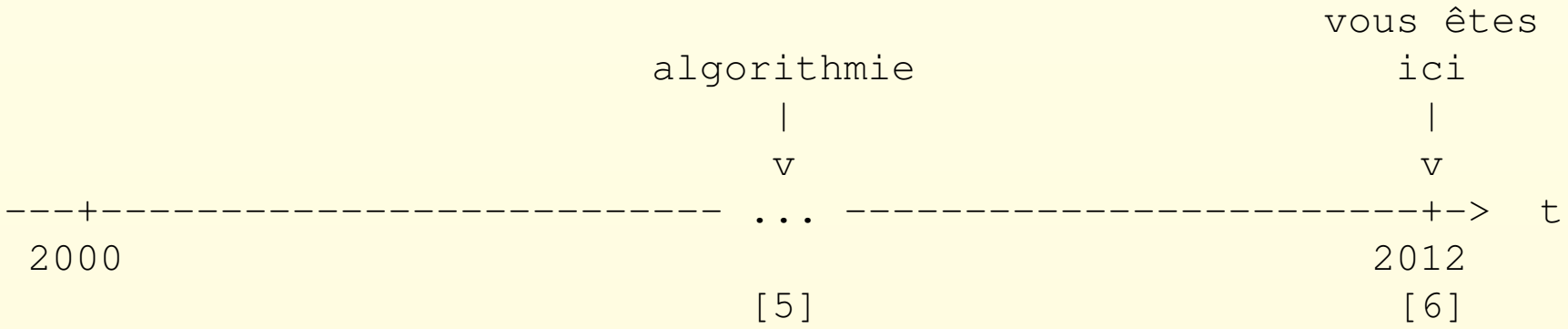
Algorithms for Mathematical Morphology. §, 2010.

[4] Préservation de l'efficacité :

Une approche générique du logiciel pour le traitement d'images préservant les performances. GRETSI, 2011.

$\longrightarrow$ un algorithme vraiment générique ne peut pas être efficace

LES TRAITEMENTS



[5] De nouveaux algorithmes :

Ruminations on Tarjan's Union-Find Algorithm and Connected Operators. ISMM, 2005.

[6] En préparation...

LES TRAITEMENTS

Quelques idées :

- On a un ensemble cohérent :
  - un catalogue types + traitements + outils, $\longrightarrow$ utilisabilité et efficience
  - les traitements tirent bénéfice du paradigme :
    - spécialisation d'algorithmes grâce aux propriétés,
    - écriture simple grâce à la généricité forte.
- Vu du côté utilisateur :

LES TRAITEMENTS

Quelques idées :

- On a un ensemble cohérent :
  - un catalogue types + traitements + outils, $\longrightarrow$ utilisabilité et efficience
  - les traitements tirent bénéfice du paradigme :
    - spécialisation d'algorithmes grâce aux propriétés,
    - écriture simple grâce à la généricité forte.
- Vu du côté utilisateur :

LES TRAITEMENTS

Quelques idées :

- On a un ensemble cohérent :
  - un catalogue types + traitements + outils, $\longrightarrow$ utilisabilité et efficience
  - les traitements tirent bénéfice du paradigme :
    - ⋆ spécialisation d'algorithmes grâce aux propriétés,
    - ⋆ écriture simple grâce à la généricité bornée. $\longrightarrow$ multi-méthodes
- Vu du côté utilisateur :

LES TRAITEMENTS

Quelques idées :

- On a un ensemble cohérent :
  - un catalogue types + traitements + outils, $\longrightarrow$ utilisabilité et efficience
  - les traitements tirent bénéfice du paradigme :
    - ★ spécialisation d'algorithmes grâce aux propriétés,
    - ★ écriture simple grâce à la généricité bornée. $\longrightarrow$ multi-méthodes
- Vu du côté utilisateur :

LES TRAITEMENTS

Quelques idées :

- On a un ensemble cohérent :
  - un catalogue types + traitements + outils, $\longrightarrow$ utilisabilité et efficience
  - les traitements tirent bénéfice du paradigme :
    - ★ spécialisation d'algorithmes grâce aux propriétés,
    - ★ écriture simple grâce à la généricité bornée. $\longrightarrow$ multi-méthodes
- Vu du côté utilisateur :
  - un traitement est une procédure,

Les traitements

Quelques idées :

- On a un ensemble cohérent :
  - un catalogue types + traitements + outils, $\longrightarrow$ utilisabilité et efficience
  - les traitements tirent bénéfice du paradigme :
    - ★ spécialisation d'algorithmes grâce aux propriétés,
    - ★ écriture simple grâce à la généricité bornée. $\longrightarrow$ multi-méthodes
- Vu du côté utilisateur :
  - un traitement est une procédure, $\longrightarrow$ une facade
  - un programme est du C amélioré. $\longrightarrow$ simplicité de l'assemblage

Les traitements

Quelques idées :

- On a un ensemble cohérent :
  - un catalogue types + traitements + outils, $\longrightarrow$ utilisabilité et efficience
  - les traitements tirent bénéfice du paradigme :
    - ★ spécialisation d'algorithmes grâce aux propriétés,
    - ★ écriture simple grâce à la généricité bornée. $\longrightarrow$ multi-méthodes
- Vu du côté utilisateur :
  - un traitement est une procédure, $\longrightarrow$ une facade
  - un programme est du C amélioré. $\longrightarrow$ simplicité de l'assemblage

Les traitements

Quelques idées :

- On a un ensemble cohérent :
  - un catalogue types + traitements + outils, $\longrightarrow$ utilisabilité et efficience
  - les traitements tirent bénéfice du paradigme :
    - ★ spécialisation d'algorithmes grâce aux propriétés,
    - ★ écriture simple grâce à la généricité bornée. $\longrightarrow$ multi-méthodes
- Vu du côté utilisateur :
  - un traitement est une procédure, $\longrightarrow$ une facade
  - un programme est du C amélioré. $\longrightarrow$ simplicité de l'assemblage

UNE BIBLIOTHÈQUE EFFECTIVE ET DISPONIBLE

- Bibliothèque MILENA: partie de la plate-forme OLÉNA

code de MILENA	109 000 lignes
cœur de MILENA	25 000 lignes

- Code ouvert et libre (licence GNU GPL v2)
  `http://olena.lrde.epita.fr`
- Vivante !
  maintenue et en évolution...

UNE BIBLIOTHÈQUE EFFECTIVE ET DISPONIBLE

- Bibliothèque MILENA: partie de la plate-forme OLÉNA

code de MILENA	109 000 lignes
cœur de MILENA	25 000 lignes

- Code ouvert et libre (licence GNU GPL v2)

  `http://olena.lrde.epita.fr`

- Vivante !
  maintenue et en évolution...

UNE BIBLIOTHÈQUE EFFECTIVE ET DISPONIBLE

- Bibliothèque MILENA: partie de la plate-forme OLÉNA

code de MILENA	109 000 lignes
cœur de MILENA	25 000 lignes

- Code ouvert et libre (licence GNU GPL v2)

  `http://olena.lrde.epita.fr`

- Vivante !
  maintenue et en évolution...

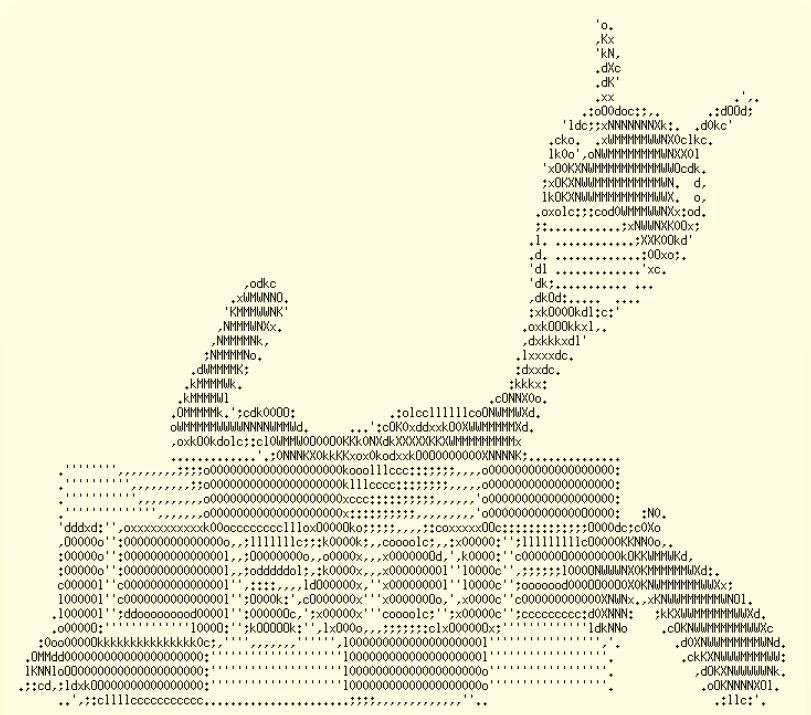
La Bibliothèque = des contributeurs

Les contributeurs :

- permanents,
- thésards, post-doctorants,
- et étudiants-chercheurs.

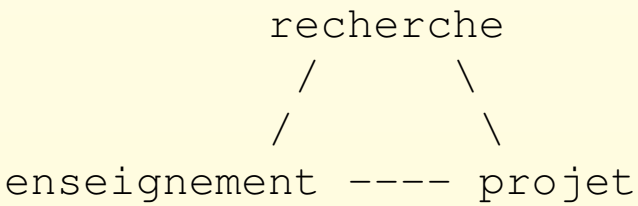
Akim Demaille — Alexandre Abraham — Alexandre Duret-Lutz — Anthony Pinagot — Astrid Wang — Benjamin Raynal
Benoît Sigoure — Caroline Vigouroux — Clément Faure — Coddy Lévi — Christophe Berger — Christopher Chedeau
Damien Thivolle — David Lesage — Dimitri Papadopoulos-Orfanos — Edwin Carlinet — Emmanuel Turquin — Etienne Folio
Fabien Freling — Francis Maes — Frédéric Bour — Geoffroy Fouquier — Giovanni Palma — Guillaume Duhamel
Guillaume Lazzara — Heru Xue — Ignacy Gawedzki — Jean Chalard — Jean-Baptiste Mouret — Jean-Sébastien Mouret
Jérôme Darbon — Johan Seland — Julia Faurie — Kristoffer Gleditsch — Loïc Fosse — Ludovic Perrine — Matthieu Garrigues
Michaël Strauss — Nicolas Ballas — Nicolas Burrus — Nicolas Pouillard — Nicolas Tisserand — Nicolas Widynski
Niels Van Vliet — Pierre-Yves Strub — Quentin Hocquet — Quôc Peyrot — Raphaël Boissel — Raphaël Poss
Renaud François — Roland Levillain — Réda Dehak — Rémi Coupet — Sébastien Crozet — Simon Odou — Simon Nivault
Sylvain Berlemont — Sylvain Lobry — Thomas Kielbus — Thomas Moulard — Tristan Croiset — Ugo Jardonnet
Vincent Berruchon — Vivien Delmon — Yann Jacquelet — Yann Régis-Gianas — Yoann Fabre — Yongchao Xu

La Bibliothèque = un environnement



SCHÉMATIQUEMENT

Le triangle d'activités :

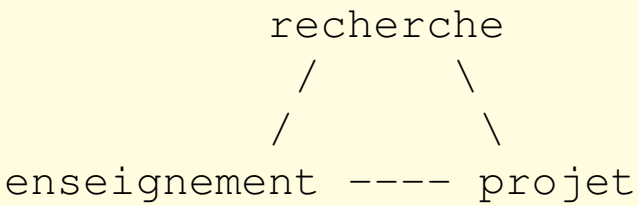


La bibliothèque touche ses 3 sommets :

- sujet de recherche en soi,
- support d'enseignement, $\longrightarrow$ utilisabilité
- valorisée dans le cadre de projets et contrats.

SCHÉMATIQUEMENT

Le triangle d'activités :

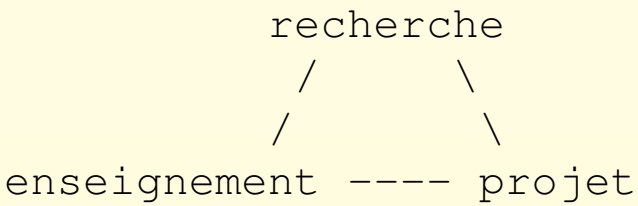


La bibliothèque touche ses 3 sommets :

- sujet de recherche en soi,
- support d'enseignement, $\longrightarrow$ utilisabilité
- valorisée dans le cadre de projets et contrats.

SCHÉMATIQUEMENT

Le triangle d'activités :



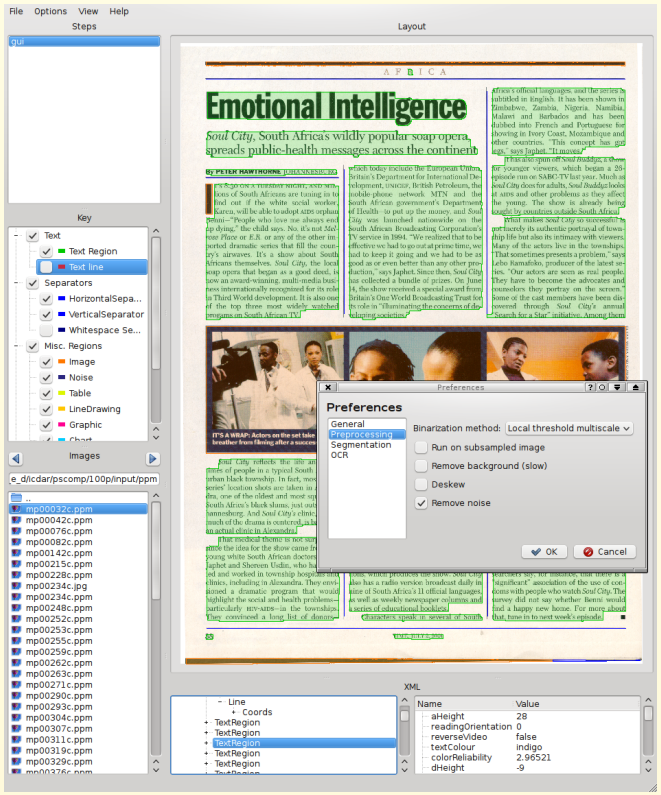
La bibliothèque touche ses 3 sommets :

- sujet de recherche en soi,
- support d'enseignement, $\longrightarrow$ utilisabilité
- valorisée dans le cadre de projets et contrats.

Recherche appliquée

- Différents domaines :
  - médical (anapath, IRM 2D, IRM 2D+*t*, écho),
  - astronomie,
  - images de documents,
  - images satellitaires,
  - images naturelles,
  - des objets 3D sous forme de surfaces maillées,
  - flux de captures d'écran,
  - à court terme... flux vidéo (films et appareils mobiles).
- Différentes tâches :
  - filtrage, recalage, segmentation,
  - reconnaissance de formes.
- Différents produits :

Un exemple :



un outil libre de dématérialisation... et de recherche reproductible.

→ possibilités ET fonctionnalités !

À PROPOS DE LA RECHERCHE

Récap :

- on a montré quelques contributions,
- on n'est pas limité / bloqué techniquement, $\longrightarrow$ prototypage rapide
- mais il y a mieux...

À PROPOS DE LA RECHERCHE

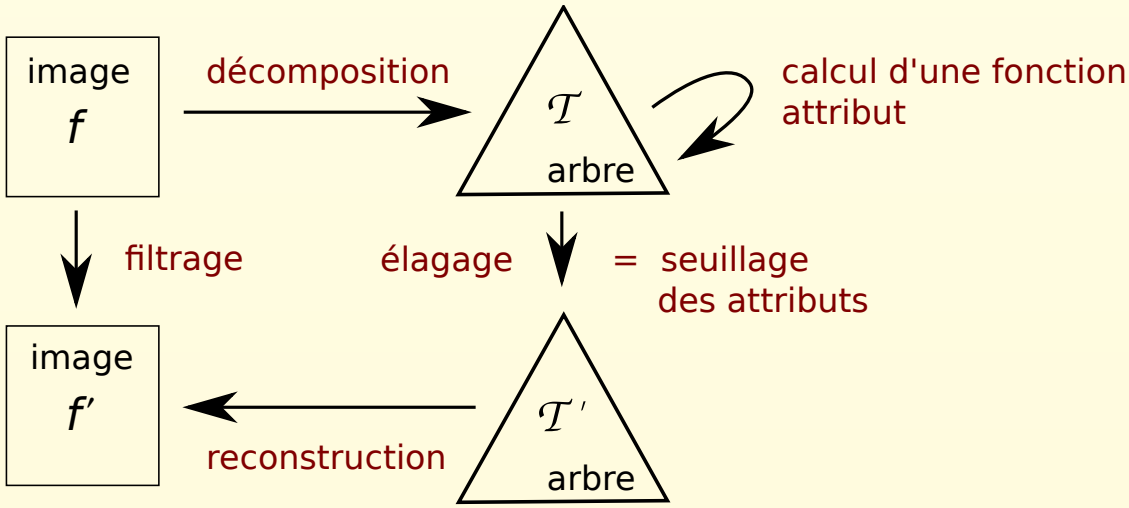
Récap :

- on a montré quelques contributions,
- on n'est pas limité / bloqué techniquement, $\longrightarrow$ prototypage rapide
- mais il y a mieux...

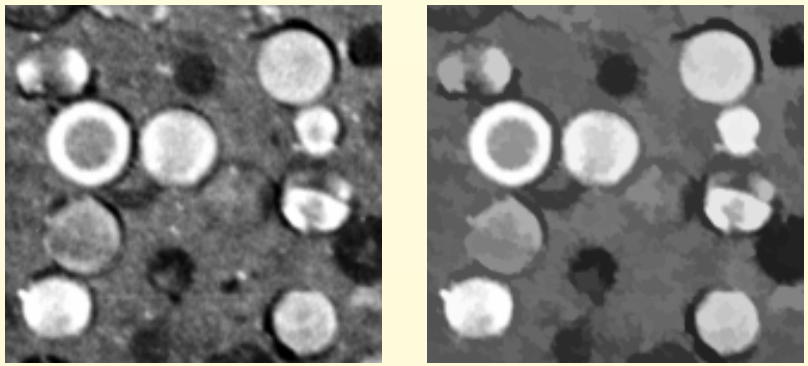
L'outil permet un déblocage psychologique !

$\longrightarrow$ facilitateur de recherche exploratoire

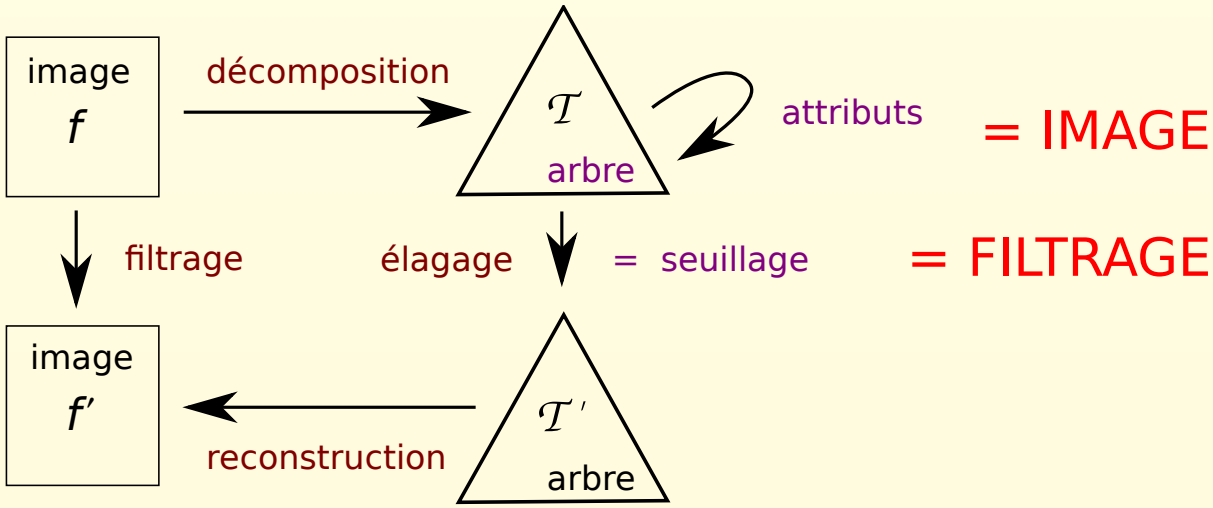
Des classes de filtres classiques :



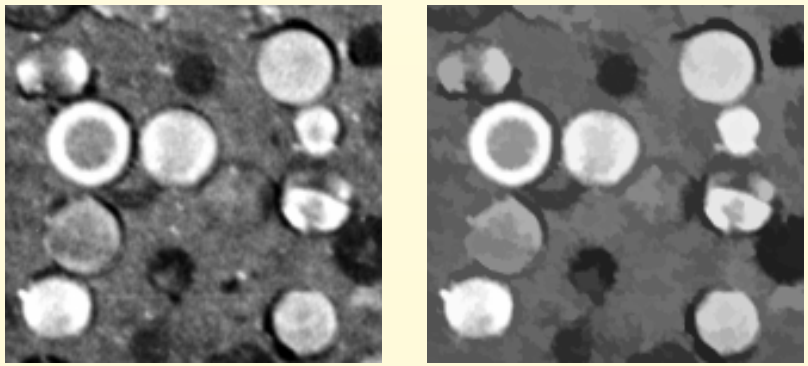
Exemple :



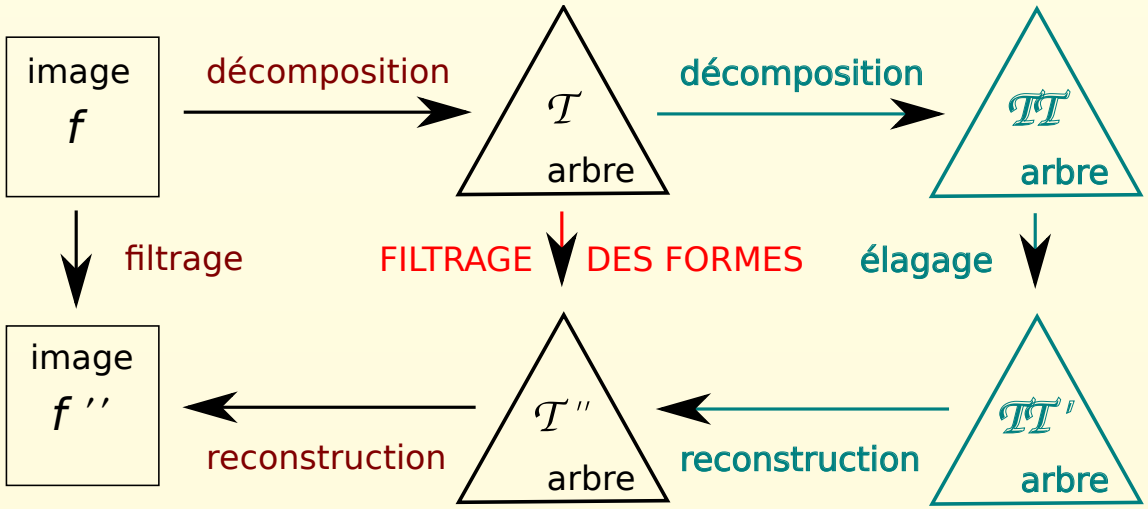
Des classes de filtres classiques :



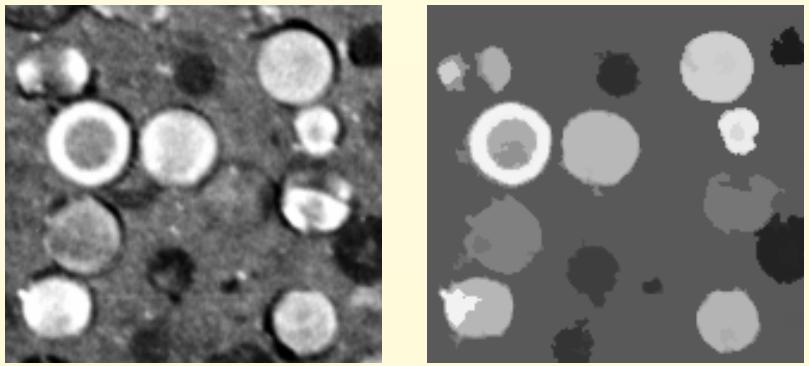
Exemple :



Une généralisation :



Exemple :



MÉTA-MESSAGE

En fait, le méta-message initial :

> Transférer certaines connaissances et offrir un outil puissant...

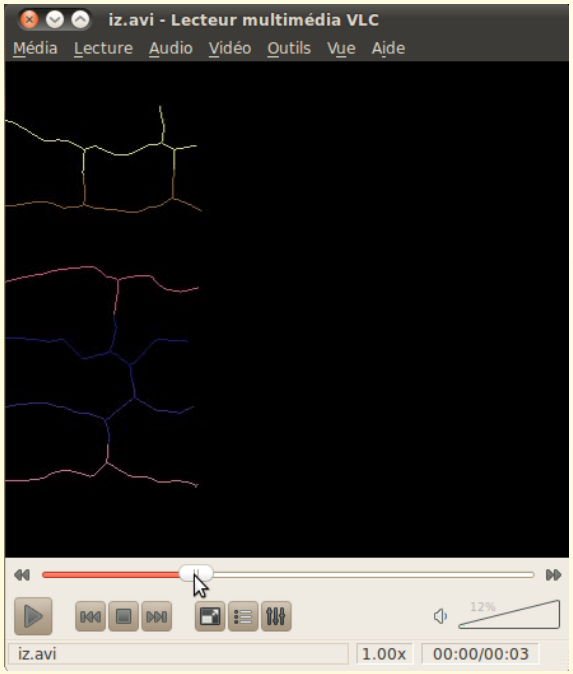
s'est traduit en :

> Garantir une utilisabilité maximale et obtenir un outil efficient.

$\longrightarrow$ on a des preuves de concepts !

L'EXEMPLE...

En vidéo :



L'EXEMPLE...

Il est effectivement illustratif :

- on n'a pas touché aux algorithmes, $\longrightarrow$ ré-utilisabilité
- le calcul d'IZ s'appuie sur un méta-algorithme, $\longrightarrow$ factorisation
- on a seulement assemblé du code, $\longrightarrow$ inter-opérabilité des composants
- "deux algorithmes = deux instructions", $\longrightarrow$ domaine métier
- l'utilisateur ne voit pas la complexité sous-jacente, $\longrightarrow$ utilisabilité
- le morpheur "magnétoscope" fait 70 lignes de code. $\longrightarrow$ efficience

L'EXEMPLE...

Il est effectivement illustratif :

- on n'a pas touché aux algorithmes, $\longrightarrow$ ré-utilisabilité
- le calcul d'IZ s'appuie sur un méta-algorithme, $\longrightarrow$ factorisation
- on a seulement assemblé du code, $\longrightarrow$ inter-opérabilité des composants
- "deux algorithmes = deux instructions", $\longrightarrow$ domaine métier
- l'utilisateur ne voit pas la complexité sous-jacente, $\longrightarrow$ utilisabilité
- le morpheur "magnétoscope" fait 70 lignes de code. $\longrightarrow$ efficience

L'EXEMPLE…

Il est effectivement illustratif :

- on n'a pas touché aux algorithmes, $\longrightarrow$ ré-utilisabilité
- le calcul d'IZ s'appuie sur un méta-algorithme, $\longrightarrow$ factorisation
- on a seulement assemblé du code, $\longrightarrow$ inter-opérabilité des composants
- "deux algorithmes = deux instructions", $\longrightarrow$ domaine métier
- l'utilisateur ne voit pas la complexité sous-jacente, $\longrightarrow$ utilisabilité
- le morpheur "magnétoscope" fait 70 lignes de code. $\longrightarrow$ efficience

L'EXEMPLE...

Il est effectivement illustratif :

- on n'a pas touché aux algorithmes, $\longrightarrow$ ré-utilisabilité
- le calcul d'IZ s'appuie sur un méta-algorithme, $\longrightarrow$ factorisation
- on a seulement assemblé du code, $\longrightarrow$ inter-opérabilité des composants
- "deux algorithmes = deux instructions", $\longrightarrow$ domaine métier
- l'utilisateur ne voit pas la complexité sous-jacente, $\longrightarrow$ utilisabilité
- le morpheur "magnétoscope" fait 70 lignes de code. $\longrightarrow$ efficience

L'EXEMPLE…

Il est effectivement illustratif :

- on n'a pas touché aux algorithmes, $\longrightarrow$ ré-utilisabilité
- le calcul d'IZ s'appuie sur un méta-algorithme, $\longrightarrow$ factorisation
- on a seulement assemblé du code, $\longrightarrow$ inter-opérabilité des composants
- "deux algorithmes = deux instructions", $\longrightarrow$ domaine métier
- l'utilisateur ne voit pas la complexité sous-jacente, $\longrightarrow$ utilisabilité
- le morpheur "magnétoscope" fait 70 lignes de code. $\longrightarrow$ efficience

L'EXEMPLE…

Il est effectivement illustratif :

- on n'a pas touché aux algorithmes, $\longrightarrow$ ré-utilisabilité
- le calcul d'IZ s'appuie sur un méta-algorithme, $\longrightarrow$ factorisation
- on a seulement assemblé du code, $\longrightarrow$ inter-opérabilité des composants
- "deux algorithmes = deux instructions", $\longrightarrow$ domaine métier
- l'utilisateur ne voit pas la complexité sous-jacente, $\longrightarrow$ utilisabilité
- le morpheur "magnétoscope" fait 70 lignes de code. $\longrightarrow$ efficience

L'EXEMPLE...

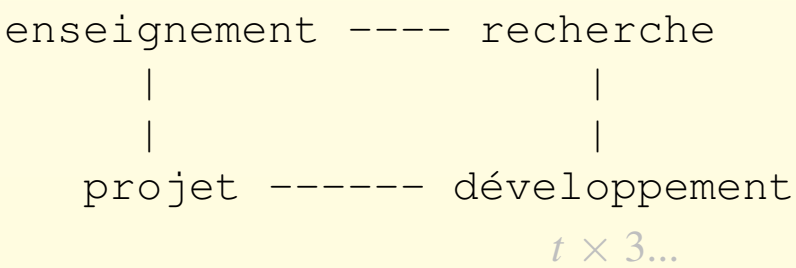
Le "vrai" code :

```
// ...

image2d<unsigned> label(input.domain());
data::fill(label, 0);

// étiquetage
unsigned nlabels;
data::paste(labeling::value(ws | make::box2d(0,0,input.nrows()-1,0),
                            0, c4(), nlabels),
            label);

// calcul des zones d'influence
data::paste(transform::influence_zone(label | (pw::value(ws) == 0),
                                      c8(), w_win2d_5_7_11),
            label);

// visualisation
output = labeling::colorize(value::rgb8(), label);
data::fill((output | pw::value(input)).rw(), literal::white);
```

Et maintenant ?

PERSPECTIVES

Un schéma moins classique :

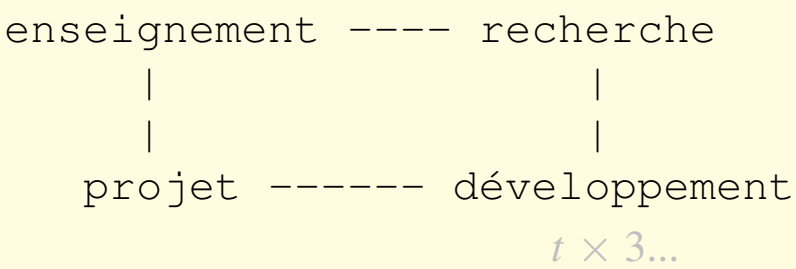


Les perspectives :

- La formalisation des abstractions, propriétés et interfaces.
- La transposition du paradigme à d'autres domaines.
- L'utilisation pour la recherche.
- ...Plus tard, la refonte.

PERSPECTIVES

Un schéma moins classique :

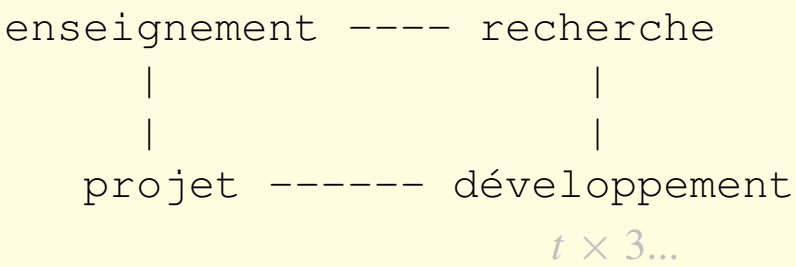


Les perspectives :

- La formalisation des abstractions, propriétés et interfaces.
- La transposition du paradigme à d'autres domaines.
- L'utilisation pour la recherche.
- ...Plus tard, la refonte.

PERSPECTIVES

Un schéma moins classique :

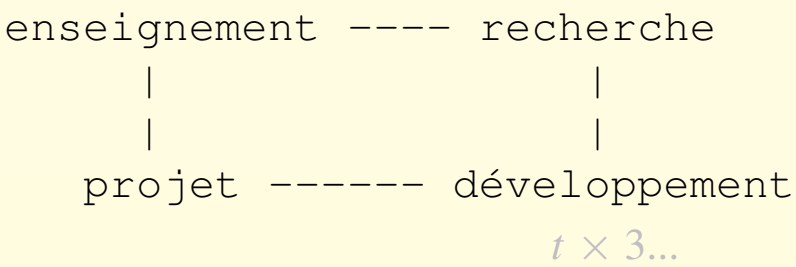


Les perspectives :

- La formalisation des abstractions, propriétés et interfaces.
- La transposition du paradigme à d'autres domaines.
- L'utilisation pour la recherche.
- ...Plus tard, la refonte.

PERSPECTIVES

Un schéma moins classique :



Les perspectives :

- La formalisation des abstractions, propriétés et interfaces.
- La transposition du paradigme à d'autres domaines.
- L'utilisation pour la recherche.
- ...Plus tard, la refonte.

Les Questions

?