Traitement d'images, logiciels, algorithmes & généricité

Jean-Baptiste Fasquel, Université d'Angers

31 Mars 2010

- Chronologie
  - 1998 : Ingénieur de l'ENSPS (Strasbourg)
  - 2002 : Docteur de l'université de Strasbourg
  - 2002-2009 : Chargé de recherche à l'IRCAD [1]
  - 2009-... : Enseignant chercheur à l'université d'Angers
- Activité
  - Traitement d'images
    - Méthodes
    - **Implémentation**
  - Applications médicales
    - **Aide au diagnostic**
    - Aide au geste thérapeutique

1. *Institut de recherche contre les cancers de l'appareil digestif, Strasbourg, France*. URL : http://www.ircad.fr/

Le traitement des images...

- Un domaine d'application vaste et complexe
  - Algorithmes de segmentation, recalage,...
  - 2D, 3D, 2D+t, 3D+t,...
  - Entrées-sorties, IHM, visualisation 3D, multithreading, ...

Le traitement des images...

- Un domaine d'application vaste et complexe
  - Algorithmes de segmentation, recalage,...
  - 2D, 3D, 2D+t, 3D+t,...
  - Entrées-sorties, IHM, visualisation 3D, multithreading, ...
- Réutilisation du code ?

Le traitement des images...

- Un domaine d'application vaste et complexe
  - Algorithmes de segmentation, recalage,...
  - 2D, 3D, 2D+t, 3D+t,...
  - Entrées-sorties, IHM, visualisation 3D, multithreading, ...
- Réutilisation du code ?
- Niveau logiciel : fonctionnalités hétérogènes
  - Algorithmes, IHM, Entrées-sorties, visualisation 3D

Le traitement des images...

- Un domaine d'application vaste et complexe
  - Algorithmes de segmentation, recalage,...
  - 2D, 3D, 2D+t, 3D+t,...
  - Entrées-sorties, IHM, visualisation 3D, multithreading, ...
- Réutilisation du code ?
- Niveau logiciel : fonctionnalités hétérogènes
  - Algorithmes, IHM, Entrées-sorties, visualisation 3D
  - Contribution: roles & composants

Le traitement des images...

- Un domaine d'application vaste et complexe
  - Algorithmes de segmentation, recalage,...
  - 2D, 3D, 2D+t, 3D+t,...
  - Entrées-sorties, IHM, visualisation 3D, multithreading, ...
- Réutilisation du code ?
- Niveau logiciel : fonctionnalités hétérogènes
  - Algorithmes, IHM, Entrées-sorties, visualisation 3D
  - Contribution: roles & composants
- Niveau algorithmique
  - Traitement bas-niveau des données

Le traitement des images...

- Un domaine d'application vaste et complexe
  - Algorithmes de segmentation, recalage,...
  - 2D, 3D, 2D+t, 3D+t,...
  - Entrées-sorties, IHM, visualisation 3D, multithreading, ...
- Réutilisation du code ?
- Niveau logiciel : fonctionnalités hétérogènes
  - Algorithmes, IHM, Entrées-sorties, visualisation 3D
  - Contribution: roles & composants
- Niveau algorithmique
  - Traitement bas-niveau des données
  - Contribution: programmation générique et régions d'intérêt

Le traitement des images...

- Un domaine d'application vaste et complexe
  - Algorithmes de segmentation, recalage,...
  - 2D, 3D, 2D+t, 3D+t,...
  - Entrées-sorties, IHM, visualisation 3D, multithreading, ...
- Réutilisation du code ?
- Niveau logiciel : fonctionnalités hétérogènes
  - Algorithmes, IHM, Entrées-sorties, visualisation 3D
  - Contribution: roles & composants
- Niveau algorithmique
  - Traitement bas-niveau des données
  - Contribution: programmation générique et régions d'intérêt
- Illustrations: applications médicales

1. Logiciels
   - Contexte
   - Contribution: roles & composants
   - Illustration
2. Algorithmes
   - Contexte
   - Contribution: régions d'intérêt (ROI)
   - Illustration
3. Synthèse

1 Logiciels
- Contexte
- Contribution: roles & composants
- Illustration

2 Algorithmes

3 Synthèse

Logiciels de traitement d'images

- Flexibilité souvent limitée aux algorithmes (plugins)
  - 3D Slicer, MevisLab, 3DVPM[2],...
  - Un programme spécifique par logiciel (*"glue code"*)

Exemple

Difficultés

2. J.-B. FASQUEL et al. "A modular and evolutive component oriented software architecture for patient modeling". Dans : *Computer Methods and Programs in Biomedicine* 83 (2006), p. 222–233.

Logiciels de traitement d'images

- Flexibilité souvent limitée aux algorithmes (plugins)
  - 3D Slicer, MevisLab, 3DVPM[2],...
  - Un programme spécifique par logiciel (*"glue code"*)

Exemple

```
#include "Image.hpp"
...
Object *obj      = new Image;
IReader *reader  = new myReader();
IRender *render  = new myRender();
...
```

Difficultés

```
...
...
Types spécifiques
Dépendance compilation
...
...
```

2. J.-B. FASQUEL et al. "A modular and evolutive component oriented software architecture for patient modeling". Dans : *Computer Methods and Programs in Biomedicine* 83 (2006), p. 222–233.

Logiciels de traitement d'images

- Flexibilité souvent limitée aux algorithmes (plugins)
  - 3D Slicer, MevisLab, 3DVPM[2],...
  - Un programme spécifique par logiciel (*"glue code"*)

Exemple

```
#include "Image.hpp"
...
Object *obj      = new Image;
IReader *reader  = new myReader();
IRender *render  = new myRender();
...
reader->setFilename(...);
...
```

Difficultés

```
...
...
Types spécifiques
Dépendance compilation
...
...
API spécifique
...
```

2. J.-B. FASQUEL et al. "A modular and evolutive component oriented software architecture for patient modeling". Dans : *Computer Methods and Programs in Biomedicine* 83 (2006), p. 222–233.

Logiciels de traitement d'images

- Flexibilité souvent limitée aux algorithmes (plugins)
  - 3D Slicer, MevisLab, 3DVPM[2],...
  - Un programme spécifique par logiciel (*"glue code"*)

Exemple

```
#include "Image.hpp"
...
Object *obj      = new Image;
IReader *reader  = new myReader();
IRender *render  = new myRender();
...
reader->setFilename(...);
...
reader->reader(obj);
render->render(obj);
...
```

Difficultés

```
...
...
Types spécifiques
Dépendance compilation
...
...
API spécifique
...
Associations explicites
Chaînage explicite
...
```

2. J.-B. FASQUEL et al. "A modular and evolutive component oriented software architecture for patient modeling". Dans : *Computer Methods and Programs in Biomedicine* 83 (2006), p. 222–233.

Logiciels de traitement d'images

- Flexibilité souvent limitée aux algorithmes (plugins)
  - 3D Slicer, MevisLab, 3DVPM[2],...
  - Un programme spécifique par logiciel (*"glue code"*)

Exemple

```
#include "Image.hpp"
...
Object *obj      = new Image;
IReader *reader  = new myReader();
IRender *render  = new myRender();
...
reader->setFilename(...);
...
reader->reader(obj);
render->render(obj);
...
//ou rafraîchissement implicite par observation?
...
```

Difficultés

```
...
...
Types spécifiques
Dépendance compilation
...
...
API spécifique
...
Associations explicites
Chaînage explicite
...
//ou Chaînage implicite?
...
```

2. J.-B. FASQUEL et al. "A modular and evolutive component oriented software architecture for patient modeling". Dans : *Computer Methods and Programs in Biomedicine* 83 (2006), p. 222–233.

Logiciels de traitement d'images

- Flexibilité souvent limitée aux algorithmes (plugins)
  - 3D Slicer, MevisLab, 3DVPM[2],...
  - Un programme spécifique par logiciel (*"glue code"*)

Exemple

```
#include "Image.hpp"
...
Object *obj      = new Image;
IReader *reader  = new myReader();
IRender *render  = new myRender();
...
reader->setFilename(...);
...
reader->reader(obj);
render->render(obj);
...
//ou rafraîchissement implicite par observation?
...
delete reader;
delete render;
delete obj;
```

Difficultés

```
...
...
Types spécifiques
Dépendance compilation
...
...
API spécifique
...
Associations explicites
Chaînage explicite
...
//ou Chaînage implicite?
...
Gestion des destructions
...
...
```

2. J.-B. FASQUEL et al. "A modular and evolutive component oriented software architecture for patient modeling". Dans : *Computer Methods and Programs in Biomedicine* 83 (2006), p. 222–233.

Vue d'ensemble

Vue d'ensemble

- Ouverture opensource en Septembre 2009 par l'IRCAD
  - fw4spl: FrameWork for Software Product Line
  - http://code.google.com/p/fw4spl/

Vue d'ensemble

- Ouverture opensource en Septembre 2009 par l'IRCAD
  - fw4spl: FrameWork for Software Product Line
  - http://code.google.com/p/fw4spl/
- L'objectif
  - Lanceur générique et minimaliste
  - Un fichier de configuration décrivant le logiciel
    - Données, fonctionnalités et leur composition

Vue d'ensemble

- Ouverture opensource en Septembre 2009 par l'IRCAD
  - fw4spl: FrameWork for Software Product Line
  - http://code.google.com/p/fw4spl/
- L'objectif
  - Lanceur générique et minimaliste
  - Un fichier de configuration décrivant le logiciel
    - Données, fonctionnalités et leur composition
- Le principe

Vue d'ensemble

- Ouverture opensource en Septembre 2009 par l'IRCAD
  - fw4spl: FrameWork for Software Product Line
  - http://code.google.com/p/fw4spl/
- L'objectif
  - Lanceur générique et minimaliste
  - Un fichier de configuration décrivant le logiciel
    - Données, fonctionnalités et leur composition
- Le principe
  - *Role: ajout (dynamique) de méthodes à un objet*
    - Meilleure structuration des éléments de code

Vue d'ensemble

- Ouverture opensource en Septembre 2009 par l'IRCAD
  - fw4spl: FrameWork for Software Product Line
  - http://code.google.com/p/fw4spl/
- L'objectif
  - Lanceur générique et minimaliste
  - Un fichier de configuration décrivant le logiciel
    - Données, fonctionnalités et leur composition
- Le principe
  - *Role: ajout (dynamique) de méthodes à un objet*
    - Meilleure structuration des éléments de code
  - *Composant: une méthode est un "plugin"*
    - Limitation des dépendances (e.g. compilation)

Objets & roles

- Une seule entité logique: *le sujet*
  - Composition/Destruction implicite
- Exemple: lecture d'une image.

Cas classique
Deux entités: une image & un lecteur
Gestion explicite de l'image et du lecteur

Architecture proposée
Une entité: le sujet = image + lecteur
"L'image prend, dynamiquement, le role de lecture"

Objets & roles

- Une seule entité logique: *le sujet*
  - Composition/Destruction implicite
- Exemple: lecture d'une image.

Cas classique
Deux entités: une image & un lecteur
Gestion explicite de l'image et du lecteur

```
//Image
Object *obj = New("Image");
...
```

Architecture proposée
Une entité: le sujet = image + lecteur
"L'image prend, dynamiquement, le role de lecture"

```
//Construction
Object::Pointer obj = New("Image");
...
```

Objets & roles

- Une seule entité logique: *le sujet*
  - Composition/Destruction implicite
- Exemple: lecture d'une image.

Cas classique
Deux entités: une image & un lecteur
Gestion explicite de l'image et du lecteur

```
//Image
Object *obj = New("Image");
...
//Lecteur
IReader *reader = new myReader();
...
```

Architecture proposée
Une entité: le sujet = image + lecteur
"L'image prend, dynamiquement, le role de lecture"

```
//Construction
Object::Pointer obj = New("Image");
...
//Lecteur
add(obj,"IReader","myReader");
...
```

Objets & roles

- Une seule entité logique: *le sujet*
  - Composition/Destruction implicite
- Exemple: lecture d'une image.

Cas classique
Deux entités: une image & un lecteur
Gestion explicite de l'image et du lecteur

```
//Image
Object *obj = New("Image");
...
//Lecteur
IReader *reader = new myReader();
...
//Lecture
reader->read(obj);
...
```

Architecture proposée
Une entité: le sujet = image + lecteur
"L'image prend, dynamiquement, le role de lecture"

```
//Construction
Object::Pointer obj = New("Image");
...
//Lecteur
add(obj,"IReader","myReader");
...
//Lecture
get(obj,"IReader")->update();
...
```

Objets & roles

- Une seule entité logique: *le sujet*
  - Composition/Destruction implicite
- Exemple: lecture d'une image.

Cas classique
Deux entités: une image & un lecteur
Gestion explicite de l'image et du lecteur

```
//Image
Object *obj = New("Image");
...
//Lecteur
IReader *reader = new myReader();
...
//Lecture
reader->read(obj);
...
//Destruction
delete obj;
delete reader;
```

Architecture proposée
Une entité: le sujet = image + lecteur
"L'image prend, dynamiquement, le role de lecture"

```
//Construction
Object::Pointer obj = New("Image");
...
//Lecteur
add(obj,"IReader","myReader");
...
//Lecture
get(obj,"IReader")->update();
...
//Destruction: obj SEULEMENT
obj->Delete();
```

Objets, Roles & Composants

- Les roles deviennent des ”plugins”: dépendances limitées

Objets, Roles & Composants

- Les roles deviennent des ”plugins”: dépendances limitées
  - Abstraction des sujets/roles: API limitée

Abstraction des sujets/roles

Sujet: édition
`add/get/erase`
Sujet: vérification
`has/support`
Role: configuration
`setConfig(...)`
Role: activation/desactivation
`start()/stop()`
Role: exécution explicite
`update()`
Role: exécution implicite (observation)
`update(messsage)`

Objets, Roles & Composants

- Les roles deviennent des ”plugins”: dépendances limitées
  - Abstraction des sujets/roles: API limitée
  - Notion de Paquet (”Bundle”)

Abstraction des sujets/roles

Sujet: édition
`add/get/erase`
Sujet: vérification
`has/support`
Role: configuration
`setConfig(...)`
Role: activation/desactivation
`start()/stop()`
Role: exécution explicite
`update()`
Role: exécution implicite (observation)
`update(messsage)`

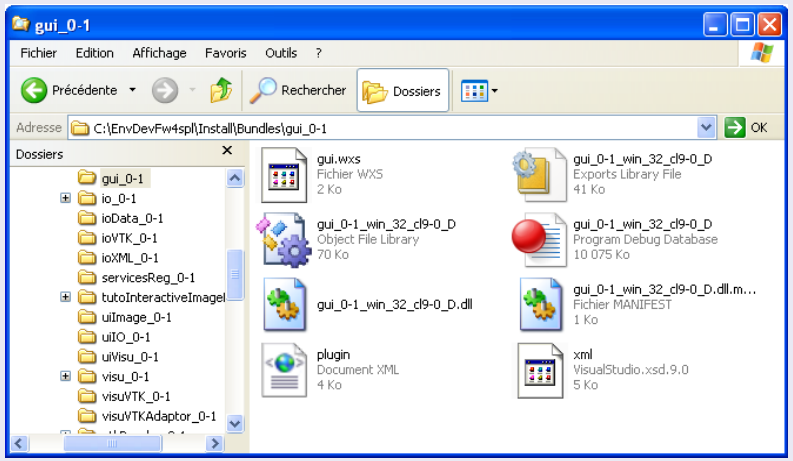
Objets, Roles & Composants

- Les roles deviennent des ”plugins”: dépendances limitées
  - Abstraction des sujets/roles: API limitée
  - Notion de Paquet (”Bundle”)
    - Un fichier de description (XML)
    - Une librairie dynamique (implémentation d'un (des) role(s))
    - Des ressources (schéma de vérification, ...)

Abstraction des sujets/roles

Sujet: édition
`add/get/erase`
Sujet: vérification
`has/support`
Role: configuration
`setConfig(...)`
Role: activation/desactivation
`start()/stop()`
Role: exécution explicite
`update()`
Role: exécution implicite (observation)
`update(messsage)`

Structure physique d'un composant/bundle (répertoire)



Objets, Roles & Composants

Description (fichier XML)

Objets, Roles & Composants

Description (fichier XML)

```
<plugin id="myComposant">




</plugin>
```

Objets, Roles & Composants

Description (fichier XML)

```
<plugin id="myComposant">
  <librairie id="libmyReader"/>




</plugin>
```

Objets, Roles & Composants

Description (fichier XML)

```
<plugin id="myComposant">
  <librairie id="libmyReader"/>
  ...
  <point id=" myReader" ...>


  </point>
  ...
</plugin>
```

Objets, Roles & Composants

Description (fichier XML)

```
<plugin id="myComposant">
  <librairie id="libmyReader"/>
  ...
  <point id=" myReader" ...>
     <implements> IReader</implements>

  </point>
  ...
</plugin>
```

Objets, Roles & Composants

Description (fichier XML)

```
<plugin id="myComposant">
  <librairie id="libmyReader"/>
  ...
  <point id=" myReader" ...>
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
  ...
</plugin>
```

Objets, Roles & Composants

Description (fichier XML)
myReader peut jouer le role d'un lecteur (**IReader**) d'image (**Image**)

```
<plugin id="myComposant">
  <librairie id="libmyReader"/>
  ...
  <point id=" myReader" ...>
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
  ...
</plugin>
```

Objets, Roles & Composants

Description (fichier XML)
myReader peut jouer le role d'un lecteur (**IReader**) d'image (**Image**)

```
<plugin id="myComposant">
  <librairie id="libmyReader"/>
  ...
  <point id=" myReader" ...>
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
  ...
</plugin>
```

Exemple de code d'invocation...

Objets, Roles & Composants

Description (fichier XML)
myReader peut jouer le role d'un lecteur (**IReader**) d'image (**Image**)

```
<plugin id="myComposant">
  <librairie id="libmyReader"/>
  ...
  <point id=" myReader" ...>
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
  ...
</plugin>
```

Exemple de code d'invocation...

```
Object::Pointer obj = New( "Image");
...
```

Objets, Roles & Composants

Description (fichier XML)
myReader peut jouer le role d'un lecteur (**IReader**) d'image (**Image**)

```
<plugin id="myComposant">
  <librairie id="libmyReader"/>
  ...
  <point id=" myReader" ...>
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
  ...
</plugin>
```

Exemple de code d'invocation...

```
Object::Pointer obj = New( "Image");
...
//Chargement libmyReader
//Instanciation de myReader
//Attachement à obj
add(obj, "IReader", "myReader");
```

Objets, Roles & Composants

Description (fichier XML)
myReader peut jouer le role d'un lecteur (**IReader**) d'image (**Image**)

```
<plugin id="myComposant">
  <librairie id="libmyReader"/>
  ...
  <point id=" myReader" schema=" myReader.xsd">
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
  ...
</plugin>
```

Exemple de code d'invocation...

```
Object::Pointer obj = New( "Image");
...
//Chargement libmyReader
//Instanciation de myReader
//Attachement à obj
add(obj, "IReader", "myReader");

//Configuration
//Vérification: myReader.xsd
get(obj, "IReader")->setConfig(...);
...
```

Objets, Roles & Composants

Description (fichier XML)
myReader peut jouer le role d'un lecteur (**IReader**) d'image (**Image**)

```
<plugin id="myComposant">
  <librairie name="libmyReader"/>
  ...
  <point id=" myReader" schema=" myReader.xsd">
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
  ...
</plugin>
```

Code sans dépendance de compilation...

```
Object::Pointer obj = New( "Image");
...
//Chargement libmyReader
//Instanciation de myReader
//Attachement à obj
add(obj, "IReader", "myReader");

//Configuration
//Vérification: myReader.xsd
get(obj, "IReader")->setConfig(...);
...
```

Un langage de composition et de configuration $\longrightarrow$ un simple interpréteur suffit: lanceur générique

Objets, Roles & Composants

Description (fichier XML)
myReader peut jouer le role d'un lecteur (**IReader**) d'image (**Image**)

```
<plugin id="myComposant">
  <librairie name="libmyReader"/>
  ...
  <point id=" myReader" schema=" myReader.xsd">
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
  ...
</plugin>
```

Code sans dépendance de compilation...

```
Object::Pointer obj = New( "Image");
...
//Chargement libmyReader
//Instanciation de myReader
//Attachement à obj
add(obj, "IReader", "myReader");

//Configuration
//Vérification: myReader.xsd
get(obj, "IReader")->setConfig(...);
...
```

Un langage de composition et de configuration $\longrightarrow$ un simple interpréteur suffit: lanceur générique

```
< object type="Image">



</ object>
```

Objets, Roles & Composants

Description (fichier XML)
myReader peut jouer le role d'un lecteur (**IReader**) d'image (**Image**)

```
<plugin id="myComposant">
  <librairie name="libmyReader"/>
  ...
  <point id=" myReader" schema=" myReader.xsd">
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
  ...
</plugin>
```

Code sans dépendance de compilation...

```
Object::Pointer obj = New( "Image");
...
//Chargement libmyReader
//Instanciation de myReader
//Attachement à obj
add(obj, "IReader", "myReader");

//Configuration
//Vérification: myReader.xsd
get(obj, "IReader")->setConfig(...);
...
```

Un langage de composition et de configuration $\longrightarrow$ un simple interpréteur suffit: lanceur générique

```
< object type="Image">
  < service type= "IReader" implementation= "myReader" >

  </ service>
</ object>
```

Objets, Roles & Composants

Description (fichier XML)
myReader peut jouer le role d'un lecteur (**IReader**) d'image (**Image**)

```
<plugin id="myComposant">
  <librairie name="libmyReader"/>
  ...
  <point id=" myReader" schema=" myReader.xsd">
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
  ...
</plugin>
```

Code sans dépendance de compilation...

```
Object::Pointer obj = New( "Image");
...
//Chargement libmyReader
//Instanciation de myReader
//Attachement à obj
add(obj, "IReader", "myReader");

//Configuration
//Vérification: myReader.xsd
get(obj, "IReader")->setConfig(...);
...
```

Un langage de composition et de configuration → un simple interpréteur suffit: lanceur générique

```
< object type="Image">
  < service type= "IReader" implementation= "myReader" >
      <filename id="monFichierImage"/> <!-- vérifié par myReader.xsd -->
  </ service>
</ object>
```

Objets, Roles, Composants & Logiciels

Fabrique de logiciels: logiciel=launcher(XML)
Exemple: cas d'un seul sujet

```
< object id="Exemple" type="Image">
  <service type="IReader" implementation="myReader" >
    <filename id="monFichierImage"/>
  </service>
  <service type="IAspect" implementation="myAspect" >
    <views>
      <view id="mainWindow"/>
    </views>
  </service>
  <service type="IRender" implementation="myRender" >
    <view id="mainWindow"/>
  </service>
  <start  type="IReader"/>
  <start  type="IAspect"/>
  <start  type="IRender"/>
  <update type="IReader"/>
  <stop   type="IRender"/>
  <stop   type="IAspect" />
</ object>
```

Un composant...

```
<plugin id="composantMyReader">
  <librairie name="libmyReader"/>
  <point id="myReader" schema="myReader.xsd">
    <implements>IReader</implements>
    <implements>Image</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyAspect">
  <librairie name="libmyAspect"/>
  <point id="myAspect" schema="myAspect.xsd">
    <implements>IAspect</implements>
    <implements>Object</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyRender">
  <librairie name="libmyRender"/>
  <point id="myRender" schema="myRender.xsd">
    <implements>IRender</implements>
    <implements>Image</implements>
  </point>
</plugin>
```

Objets, Roles, Composants & Logiciels

Fabrique de logiciels: logiciel=launcher(XML)
Exemple: cas d'un seul sujet

```
< object id="Exemple" type="Image">
  <service type="IReader" implementation="myReader" >
    <filename id="monFichierImage"/>
  </service>
  <service type="IAspect" implementation="myAspect" >
    <views>
      <view id="mainWindow"/>
    </views>
  </service>
  <service type="IRender" implementation="myRender" >
    <view id="mainWindow"/>
  </service>
  <start  type="IReader"/>
  <start  type="IAspect"/>
  <start  type="IRender"/>
  <update type="IReader"/>
  <stop   type="IRender"/>
  <stop   type="IAspect" />
</ object>
```

Un composant...

```
<plugin id="composantMyReader">
  <librairie name="libmyReader"/>
  <point id="myReader" schema="myReader.xsd">
     <implements>IReader</implements>
     <implements>Image</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyAspect">
  <librairie name="libmyAspect"/>
  <point id="myAspect" schema="myAspect.xsd">
     <implements>IAspect</implements>
     <implements>Object</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyRender">
  <librairie name="libmyRender"/>
  <point id="myRender" schema="myRender.xsd">
     <implements>IRender</implements>
     <implements>Image</implements>
  </point>
</plugin>
```

Objets, Roles, Composants & Logiciels

Fabrique de logiciels: logiciel=launcher(XML)
Exemple: cas d'un seul sujet

```
< object id="Exemple" type="Image">
  <service type="IReader" implementation="myReader" >
    <filename id="monFichierImage"/>
  </service>
  <service type="IAspect" implementation="myAspect" >
    <views>
      <view id="mainWindow"/>
    </views>
  </service>
  <service type="IRender" implementation="myRender" >
    <view id="mainWindow"/>
  </service>
  <start  type="IReader"/>
  <start  type="IAspect"/>
  <start  type="IRender"/>
  <update type="IReader"/>
  <stop   type="IRender"/>
  <stop   type="IAspect" />
</ object>
```

Un composant...

```
<plugin id="composantMyReader">
  <librairie name="libmyReader"/>
  <point id="myReader" schema="myReader.xsd">
     <implements>IReader</implements>
     <implements>Image</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyAspect">
  <librairie name="libmyAspect"/>
  <point id="myAspect" schema="myAspect.xsd">
     <implements>IAspect</implements>
     <implements>Object</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyRender">
  <librairie name="libmyRender"/>
  <point id="myRender" schema="myRender.xsd">
     <implements>IRender</implements>
     <implements>Image</implements>
  </point>
</plugin>
```

Objets, Roles, Composants & Logiciels

Fabrique de logiciels: logiciel=launcher(XML)
Exemple: cas d'un seul sujet

```
< object id="Exemple" type="Image">
  <service type="IReader" implementation="myReader" >
    <filename id="monFichierImage"/>
  </service>
  <service type="IAspect" implementation="myAspect" >
    <views>
      <view id=" mainWindow"/>
    </views>
  </service>
  <service type="IRender" implementation="myRender" >
    <view id=" mainWindow"/>
  </service>
  <start  type="IReader"/>
  <start  type="IAspect"/>
  <start  type="IRender"/>
  <update type="IReader"/>
  <stop   type="IRender"/>
  <stop   type="IAspect" />
</ object>
```

Un composant...

```
<plugin id="composantMyReader">
  <librairie name="libmyReader"/>
  <point id=" myReader" schema=" myReader.xsd">
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyAspect">
  <librairie name="libmyAspect"/>
  <point id=" myAspect" schema=" myAspect.xsd">
     <implements> IAspect</implements>
     <implements> Object</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyRender">
  <librairie name="libmyRender"/>
  <point id=" myRender" schema=" myRender.xsd">
     <implements> IRender</implements>
     <implements> Image</implements>
  </point>
</plugin>
```

Objets, Roles, Composants & Logiciels

Fabrique de logiciels: logiciel=launcher(XML)
Exemple: cas d'un seul sujet

```
< object id="Exemple" type="Image">
  <service type="IReader" implementation="myReader" >
    <filename id="monFichierImage"/>
  </service>
  <service type="IAspect" implementation="myAspect" >
    <views>
      <view id=" mainWindow"/>
    </views>
  </service>
  <service type="IRender" implementation="myRender" >
    <view id=" mainWindow"/>
  </service>
  <start  type="IReader"/>
  <start  type="IAspect"/>
  <start  type="IRender"/>
  <update type="IReader"/>
  <stop   type="IRender"/>
  <stop   type="IAspect" />
</ object>
```

Un composant...

```
<plugin id="composantMyReader">
  <librairie name="libmyReader"/>
  <point id=" myReader" schema=" myReader.xsd">
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyAspect">
  <librairie name="libmyAspect"/>
  <point id=" myAspect" schema=" myAspect.xsd">
     <implements> IAspect</implements>
     <implements> Object</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyRender">
  <librairie name="libmyRender"/>
  <point id=" myRender" schema=" myRender.xsd">
     <implements> IRender</implements>
     <implements> Image</implements>
  </point>
</plugin>
```

Objets, Roles, Composants & Logiciels

Fabrique de logiciels: logiciel=launcher(XML)
Exemple: cas d'un seul sujet

```
< object id="Exemple" type="Image">
  <service type="IReader" implementation="myReader" >
    <filename id="monFichierImage"/>
  </service>
  <service type="IAspect" implementation="myAspect" >
    <views>
      <view id=" mainWindow"/>
    </views>
  </service>
  <service type="IRender" implementation="myRender" >
    <view id=" mainWindow"/>
  </service>
  <start  type="IReader"/>
  <start  type="IAspect"/>
  <start  type="IRender"/>
  <update type="IReader"/>
  <stop   type="IRender"/>
  <stop   type="IAspect" />
</ object>
```

Un composant...

```
<plugin id="composantMyReader">
  <librairie name="libmyReader"/>
  <point id=" myReader" schema=" myReader.xsd">
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyAspect">
  <librairie name="libmyAspect"/>
  <point id=" myAspect" schema=" myAspect.xsd">
     <implements> IAspect</implements>
     <implements> Object</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyRender">
  <librairie name="libmyRender"/>
  <point id=" myRender" schema=" myRender.xsd">
     <implements> IRender</implements>
     <implements> Image</implements>
  </point>
</plugin>
```

Objets, Roles, Composants & Logiciels

Fabrique de logiciels: logiciel=launcher(XML)
Exemple: cas d'un seul sujet

```
< object id="Exemple" type="Image">
  <service type="IReader" implementation="myReader" >
    <filename id="monFichierImage"/>
  </service>
  <service type="IAspect" implementation="myAspect" >
    <views>
      <view id=" mainWindow"/>
    </views>
  </service>
  <service type="IRender" implementation="myRender" >
    <view id=" mainWindow"/>
  </service>
  <start  type="IReader"/>
  <start  type="IAspect"/>
  <start  type="IRender"/>
  <update type="IReader"/>
  <stop   type="IRender"/>
  <stop   type="IAspect" />
</ object>
```

Un composant...

```
<plugin id="composantMyReader">
  <librairie name="libmyReader"/>
  <point id=" myReader" schema=" myReader.xsd">
     <implements> IReader</implements>
     <implements> Image</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyAspect">
  <librairie name="libmyAspect"/>
  <point id=" myAspect" schema=" myAspect.xsd">
     <implements> IAspect</implements>
     <implements> Object</implements>
  </point>
</plugin>
```

Un composant...

```
<plugin id="composantMyRender">
  <librairie name="libmyRender"/>
  <point id=" myRender" schema=" myRender.xsd">
     <implements> IRender</implements>
     <implements> Image</implements>
  </point>
</plugin>
```

Objets, Roles, Composants & Logiciels

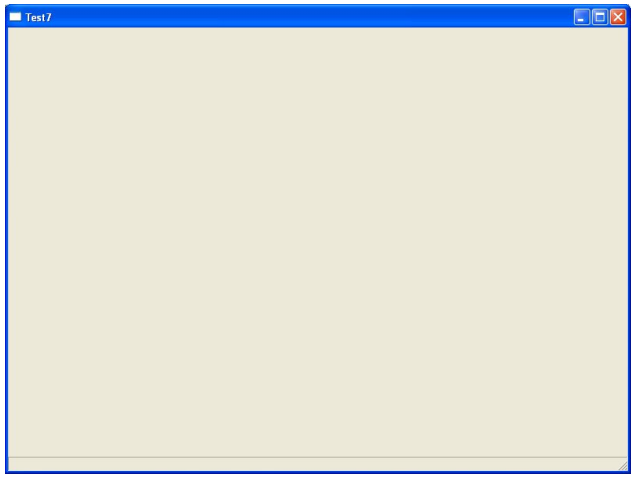
Fabrique de logiciels: logiciel=launcher(XML)
Exemple: cas d'un seul sujet

```
< object id="Exemple" type="Image">
  <service type="IReader" implementation="myReader" >
    <filename id="monFichierImage"/>
  </service>
  <service type="IAspect" implementation="myAspect" >
    <views>
      <view id="mainWindow"/>
    </views>
  </service>
  <service type="IRender" implementation="myRender" >
    <view id=" mainWindow"/>
  </service>
  <start  type="IReader"/>
  <start  type="IAspect"/>
  <start  type="IRender"/>
  <update type="IReader"/>
  <stop   type="IRender"/>
  <stop   type="IAspect" />
</ object>
```

Objets, Roles, Composants & Logiciels

Fabrique de logiciels: logiciel=launcher(XML)
Exemple: cas d'un seul sujet

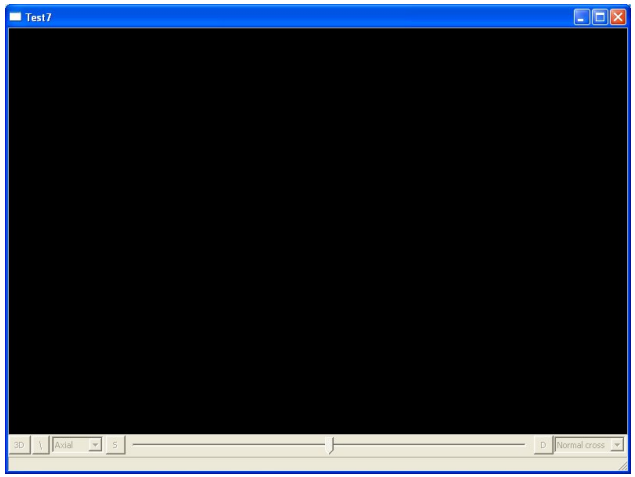
```
< object id="Exemple" type="Image">
  <service type="IReader" implementation="myReader" >
    <filename id="monFichierImage"/>
  </service>
  <service type="IAspect" implementation="myAspect" >
    <views>
      <view id="mainWindow"/>
    </views>
  </service>
  <service type="IRender" implementation="myRender" >
    <view id=" mainWindow"/>
  </service>
  <start  type="IReader"/>
  <start  type="IAspect"/>
  <start  type="IRender"/>
  <update type="IReader"/>
  <stop   type="IRender"/>
  <stop   type="IAspect" />
</ object>
```



Objets, Roles, Composants & Logiciels

Fabrique de logiciels: logiciel=launcher(XML)
Exemple: cas d'un seul sujet

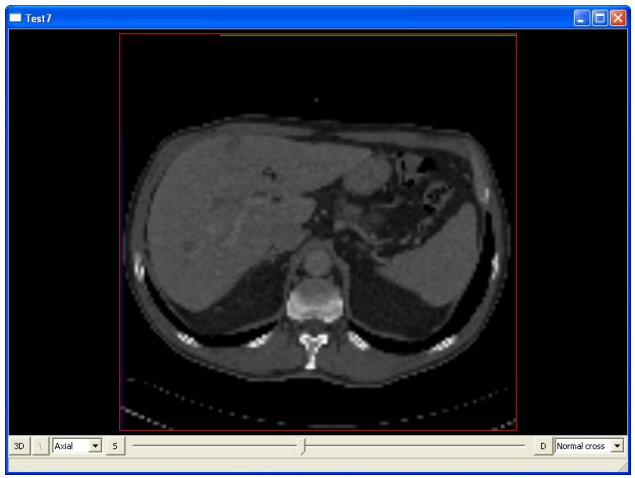
```
< object id="Exemple" type="Image">
  <service type="IReader" implementation="myReader" >
    <filename id="monFichierImage"/>
  </service>
  <service type="IAspect" implementation="myAspect" >
    <views>
      <view id="mainWindow"/>
    </views>
  </service>
  <service type="IRender" implementation="myRender" >
    <view id=" mainWindow"/>
  </service>
  <start  type="IReader"/>
  <start  type="IAspect"/>
  <start  type="IRender"/>
  <update type="IReader"/>
  <stop   type="IRender"/>
  <stop   type="IAspect" />
</ object>
```



Objets, Roles, Composants & Logiciels

Fabrique de logiciels: logiciel=launcher(XML)
Exemple: cas d'un seul sujet

```
< object id="Exemple" type="Image">
  <service type="IReader" implementation="myReader" >
    <filename id="monFichierImage"/>
  </service>
  <service type="IAspect" implementation="myAspect" >
    <views>
      <view id="mainWindow"/>
    </views>
  </service>
  <service type="IRender" implementation="myRender" >
    <view id=" mainWindow"/>
  </service>
  <start  type="IReader"/>
  <start  type="IAspect"/>
  <start  type="IRender"/>
  <update type="IReader"/>
  <stop   type="IRender"/>
  <stop   type="IAspect" />
</ object>
```

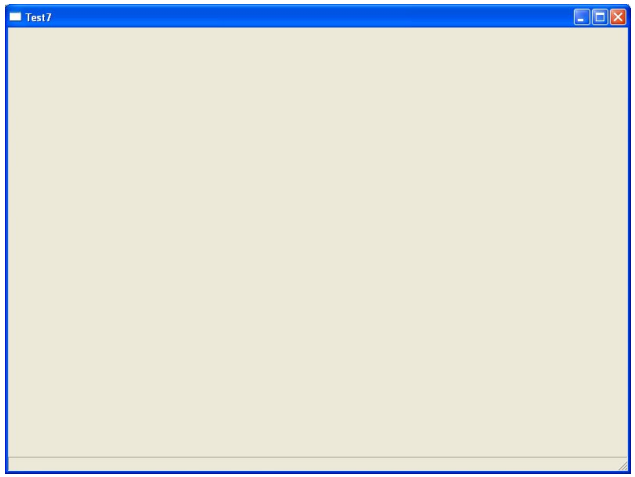


Chaînage implicite
L'implémentation du pattern Observer est...
un role!

Objets, Roles, Composants & Logiciels

Fabrique de logiciels: logiciel=launcher(XML)
Exemple: cas d'un seul sujet

```
< object id="Exemple" type="Image">
  <service type="IReader" implementation="myReader" >
    <filename id="monFichierImage"/>
  </service>
  <service type="IAspect" implementation="myAspect" >
    <views>
      <view id="mainWindow"/>
    </views>
  </service>
  <service type="IRender" implementation="myRender" >
    <view id=" mainWindow"/>
  </service>
  <start  type="IReader"/>
  <start  type="IAspect"/>
  <start  type="IRender"/>
  <update type="IReader"/>
  <stop   type="IRender"/>
  <stop   type="IAspect" />
</ object>
```



Objets, Roles, Composants & Logiciels

Fabrique de logiciels: logiciel=launcher(XML)
Exemple: cas d'un seul sujet

```
< object id="Exemple" type="Image">
  <service type="IReader" implementation="myReader" >
    <filename id="monFichierImage"/>
  </service>
  <service type="IAspect" implementation="myAspect" >
    <views>
      <view id="mainWindow"/>
    </views>
  </service>
  <service type="IRender" implementation="myRender" >
    <view id=" mainWindow"/>
  </service>
  <start  type="IReader"/>
  <start  type="IAspect"/>
  <start  type="IRender"/>
  <update type="IReader"/>
  <stop   type="IRender"/>
  <stop   type="IAspect" />
</ object>
```

Autour du traitement d'images

- Les role-composants sont
  - Algorithmes
    - Segmentation, recalage, tracking...
  - Eléments d'IHM
    - Fenêtre, menus, actions...
  - Eléments de rendu 3D
    - Primitives, Interactions, Caméra...
  - Entrèes-sorties
  - ... Données!!!!
- Quelques exemples...

Segmentation des images

Fabrique de logiciels: logiciel=launcher(XML)
BasicProcessor: cas d'un sujet composite

```
<object id="basicProcessor" type="Object">
 <object id="Input0" type="Image">
  <service type="IReader" implementation="myReader" .../>
  <service type="IRender" implementation="myRender" .../>
 </object>
 <object id="Input1" type="Float"/>
 ...
 <object id="Output0" type="Image">
  <service type="IRender" implementation="myRender" .../>
 </object>
 <service type="IProcessor" implementation="erode" >
   <Input id="Input0" .../>
    <param1 id="Input1" .../>
   ...
   <Output id="Output0" .../>
 </service>
 <service type="IAspect" implementation="myAspect" .../>
 <service type="IEditor" implementation="myEditor" .../>
    ...
</object>
```

Segmentation des images

Fabrique de logiciels: logiciel=launcher(XML)
BasicProcessor: cas d'un sujet composite

```
<object id="basicProcessor" type="Object">
 <object id="Input0" type="Image">
  <service type="IReader" implementation="myReader" .../>
  <service type="IRender" implementation="myRender" .../>
 </object>
 <object id="Input1" type="Float"/>
 ...
 <object id="Output0" type="Image">
  <service type="IRender" implementation="myRender" .../>
 </object>
 <service type="IProcessor" implementation="erode" >
   <Input id="Input0" .../>
    <param1 id="Input1" .../>
   ...
   <Output id="Output0" .../>
 </service>
 <service type="IAspect" implementation="myAspect" .../>
 <service type="IEditor" implementation="myEditor" .../>
    ...
</object>
```

Segmentation des images

Fabrique de logiciels: logiciel=launcher(XML)
BasicProcessor: cas d'un sujet composite

```
<object id="basicProcessor" type="Object">
 <object id="Input0" type="Image">
  <service type="IReader" implementation="myReader" .../>
  <service type="IRender" implementation="myRender" .../>
 </object>
 <object id="Input1" type="Float"/>
 ...
 <object id="Output0" type="Image">
  <service type="IRender" implementation="myRender" .../>
 </object>
 <service type="IProcessor" implementation="erode" >
   <Input id="Input0" .../>
   <param1 id="Input1" .../>
   ...
   <Output id="Output0" .../>
 </service>
 <service type="IAspect" implementation="myAspect" .../>
 <service type="IEditor" implementation="myEditor" .../>
    ...
</object>
```

Segmentation des images

Fabrique de logiciels: logiciel=launcher(XML)
BasicProcessor: cas d'un sujet composite

```
<object id="basicProcessor" type="Object">
 <object id="Input0" type="Image">
  <service type="IReader" implementation="myReader" .../>
  <service type="IRender" implementation="myRender" .../>
 </object>
 <object id="Input1" type="Float"/>
 ...
 <object id="Output0" type="Image">
  <service type="IRender" implementation="myRender" .../>
 </object>
 <service type="IProcessor" implementation="erode" >
   <Input id="Input0" .../>
   <param1 id="Input1" .../>
   ...
   <Output id="Output0" .../>
 </service>
 <service type="IAspect" implementation="myAspect" .../>
 <service type="IEditor" implementation="myEditor" .../>
    ...
</object>
```

Segmentation des images

Fabrique de logiciels: logiciel=launcher(XML)
BasicProcessor: cas d'un sujet composite

```
<object id="basicProcessor" type="Object">
 <object id="Input0" type="Image">
  <service type="IReader" implementation="myReader" .../>
  <service type="IRender" implementation="myRender" .../>
 </object>
 <object id="Input1" type="Float"/>
 ...
 <object id="Output0" type="Image">
  <service type="IRender" implementation="myRender" .../>
 </object>
 <service type="IProcessor" implementation="erode" >
   <Input id="Input0" .../>
   <param1 id="Input1" .../>
   ...
   <Output id="Output0" .../>
 </service>
 <service type="IAspect" implementation="myAspect" .../>
 <service type="IEditor" implementation="myEditor" .../>
    ...
</object>
```

Segmentation des images

Fabrique de logiciels: logiciel=launcher(XML)
BasicProcessor: cas d'un sujet composite

```
<object id="basicProcessor" type="Object">
 <object id=" Input0" type="Image">
  <service type="IReader" implementation="myReader" .../>
  <service type="IRender" implementation="myRender" .../>
 </object>
 <object id=" Input1" type="Float"/>
 ...
 <object id=" Output0" type="Image">
  <service type="IRender" implementation="myRender" .../>
 </object>
 <service type="IProcessor" implementation="erode" >
   <Input id=" Input0" .../>
   <param1 id=" Input1" .../>
   ...
   <Output id=" Output0" .../>
 </service>
 <service type="IAspect" implementation="myAspect" .../>
 <service type="IEditor" implementation="myEditor" .../>
    ...
</object>
```

Segmentation des images

Fabrique de logiciels: logiciel=launcher(XML)
BasicProcessor: cas d'un sujet composite

```
<object id="basicProcessor" type="Object">
 <object id=" Input0" type="Image">
  <service type="IReader" implementation="myReader" .../>
  <service type="IRender" implementation="myRender" .../>
 </object>
 <object id=" Input1" type="Float"/>
 ...
 <object id=" Output0" type="Image">
  <service type="IRender" implementation="myRender" .../>
 </object>
 <service type="IProcessor" implementation="erode" >
   <Input id=" Input0" .../>
   <param1 id=" Input1" .../>
   ...
   <Output id=" Output0" .../>
 </service>
 <service type="IAspect" implementation="myAspect" .../>
 <service type="IEditor" implementation="myEditor" .../>
    ...
</object>
```

Segmentation des images

Fabrique de logiciels: logiciel=launcher(XML)
BasicProcessor: cas d'un sujet composite

```
<object id="basicProcessor" type="Object">
 <object id=" Input0" type="Image">
  <service type="IReader" implementation="myReader" .../>
  <service type="IRender" implementation="myRender" .../>
 </object>
 <object id=" Input1" type="Float"/>
 ...
 <object id=" Output0" type="Image">
  <service type="IRender" implementation="myRender" .../>
 </object>
 <service type="IProcessor" implementation="erode" >
   <Input id=" Input0" .../>
   <param1 id=" Input1" .../>
   ...
   <Output id=" Output0" .../>
 </service>
 <service type="IAspect" implementation="myAspect" .../>
 <service type="IEditor" implementation="myEditor" .../>
     ...
</object>
```

Segmentation des images

Fabrique de logiciels: logiciel=launcher(XML)
BasicProcessor: cas d'un sujet composite

```
<object id="basicProcessor" type="Object">
 <object id=" Input0" type="Image">
  <service type="IReader" implementation="myReader" .../>
  <service type="IRender" implementation="myRender" .../>
 </object>
 <object id=" Input1" type="Float"/>
 ...
 <object id=" Output0" type="Image">
  <service type="IRender" implementation="myRender" .../>
 </object>
 <service type="IProcessor" implementation="erode" >
   <Input id=" Input0" .../>
   <param1 id=" Input1" .../>
   ...
   <Output id=" Output0" .../>
 </service>
 <service type="IAspect" implementation="myAspect" .../>
 <service type="IEditor" implementation="myEditor" .../>
     ...
</object>
```

Segmentation des images

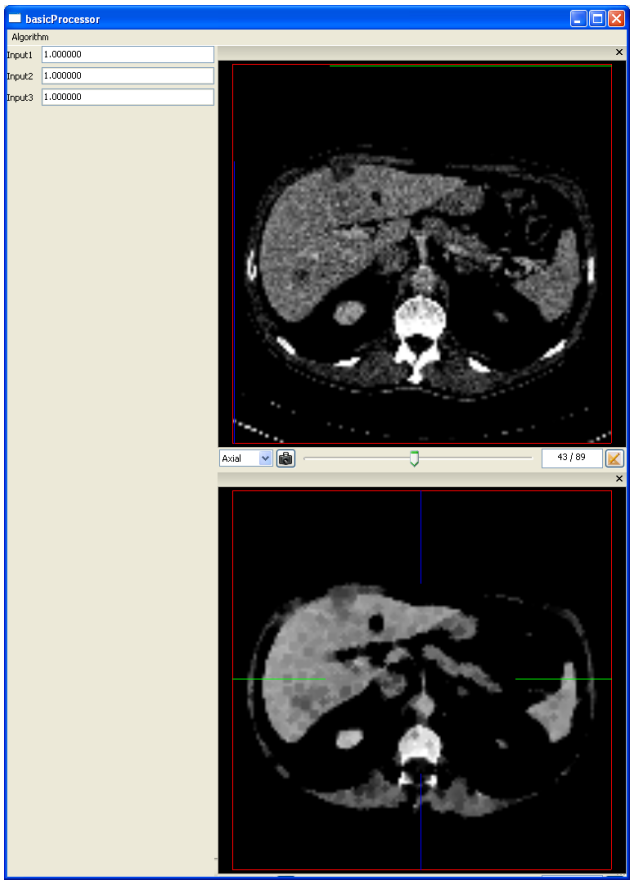
Fabrique de logiciels: logiciel=launcher(XML)
BasicProcessor: cas d'un sujet composite

```
<object id="basicProcessor" type="Object">
 <object id=" Input0" type="Image">
  <service type="IReader" implementation="myReader" .../>
  <service type="IRender" implementation="myRender" .../>
 </object>
 <object id=" Input1" type="Float"/>
 ...
 <object id=" Output0" type="Image">
  <service type="IRender" implementation="myRender" .../>
 </object>
 <service type="IProcessor" implementation="erode" >
   <Input id=" Input0" .../>
   <param1 id=" Input1" .../>
   ...
   <Output id=" Output0" .../>
 </service>
 <service type="IAspect" implementation="myAspect" .../>
 <service type="IEditor" implementation="myEditor" .../>
     ...
</object>
```

Segmentation des images

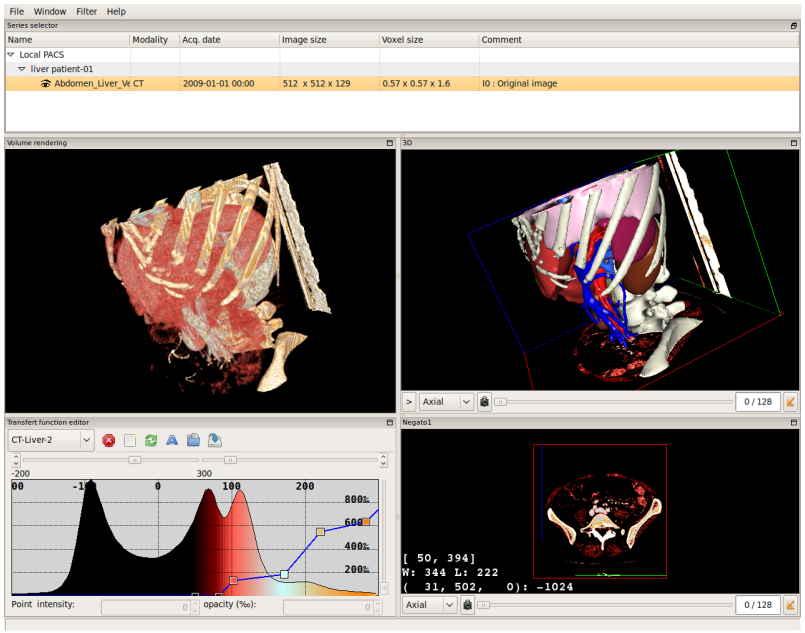
Fabrique de logiciels: logiciel=launcher(XML)
BasicProcessor: cas d'un sujet composite

```
<object id="basicProcessor" type="Object">
 <object id=" Input0" type="Image">
  <service type="IReader" implementation="myReader" .../>
  <service type="IRender" implementation="myRender" .../>
 </object>
 <object id=" Input1" type="Float"/>
 ...
 <object id=" Output0" type="Image">
  <service type="IRender" implementation="myRender" .../>
 </object>
 <service type="IProcessor" implementation="erode" >
   <Input id=" Input0" .../>
   <param1 id=" Input1" .../>
   ...
   <Output id=" Output0" .../>
 </service>
 <service type="IAspect" implementation="myAspect" .../>
 <service type="IEditor" implementation="myEditor" .../>
     ...
</object>
```



Visualisation avancée

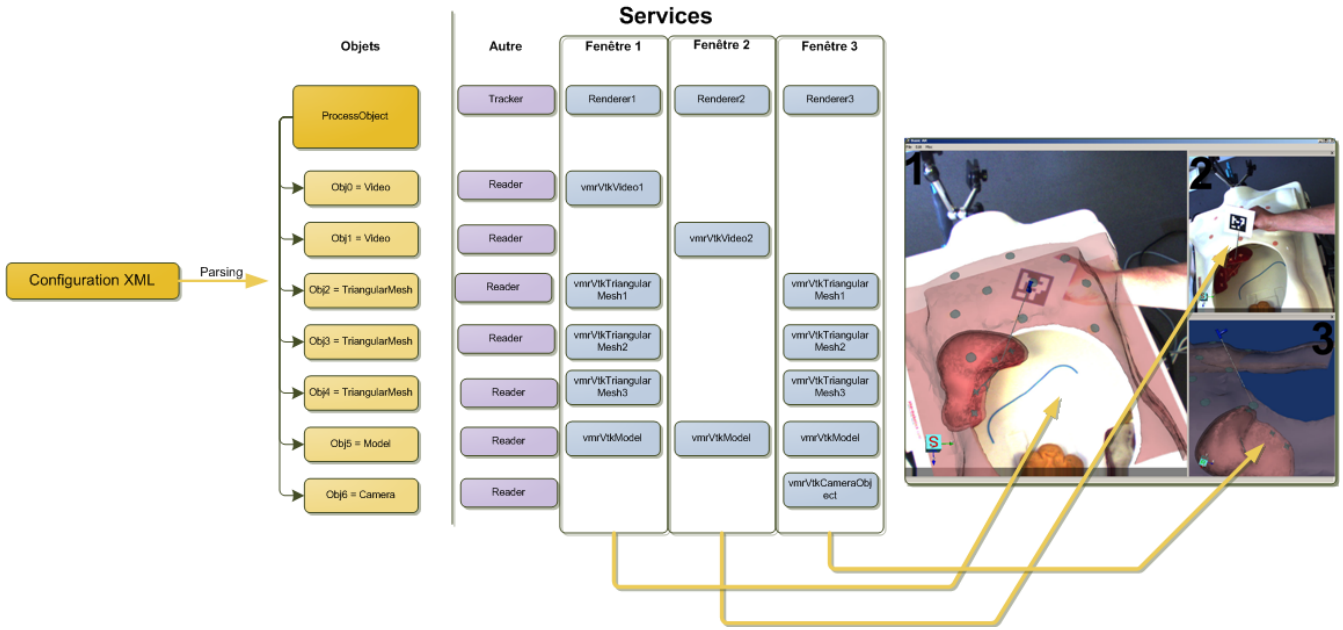
- Logiciel avancée (freeware): VR-Render[3]
  - Objets: Image/Structure 3D
  - Roles: IHM/lecteurs/sélection/visualisation



3. *VR-Render*. URL : `http://www.ircad.fr/softwares/vr-render/Software.php`

Réalité augmentée (1/2)

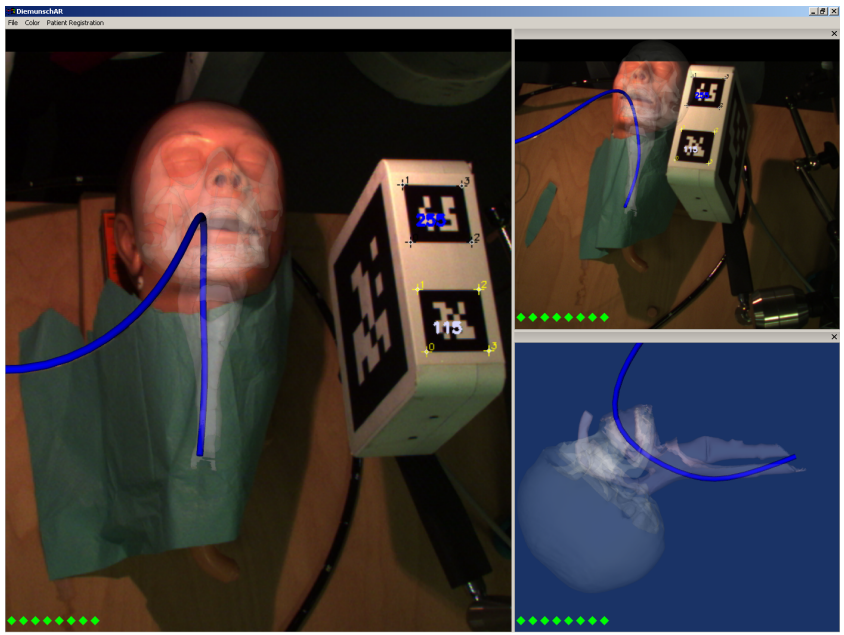
- Tracking visuel & recalage[4]
  - Objets: Video/Structure 3D
  - Roles: IHM/lecteur/tracker/visualisation



4. J.-B. FASQUEL et al. “A XML based component oriented architecture for image guided surgery: illustration for the video based tracking of a surgical tool”. Dans : *Insight Journal, Workshop on Systems and Architectures for Computer Assisted Interventions, 11th International Conference on Medical Image Computing and Computer Assisted Intervention*. 2008. URL : http://hdl.handle.net/10380/1497

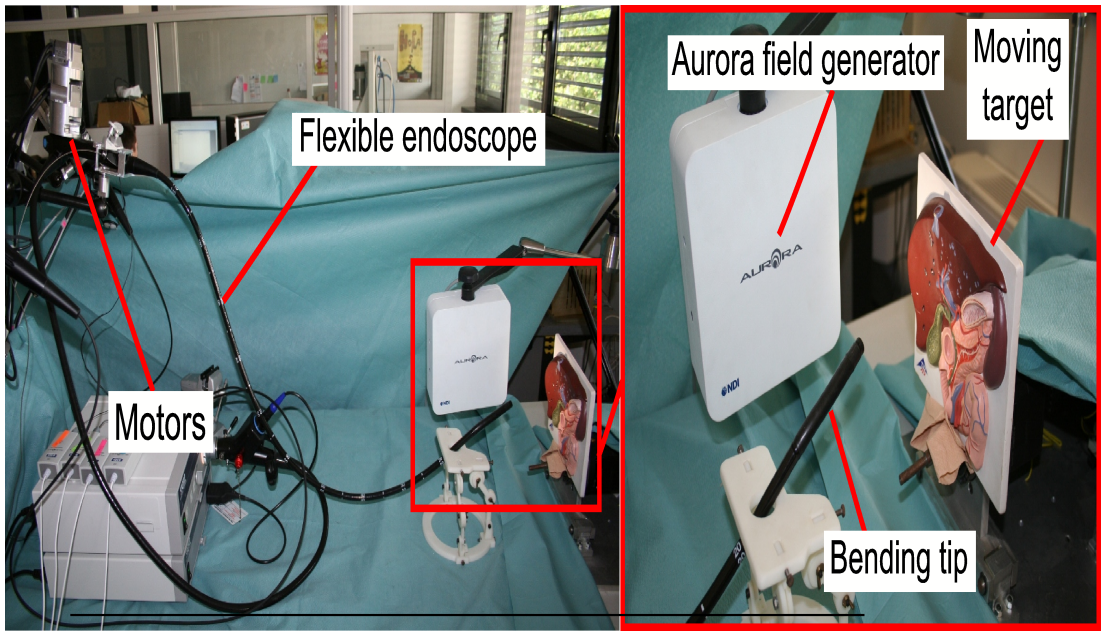
Réalité augmentée (2/2)

- Tracking électromagnétique & recalage
  - Objets: Videos/Structures 3D
  - Roles: IHM/lecteurs/tracker/visualisation



Robotique

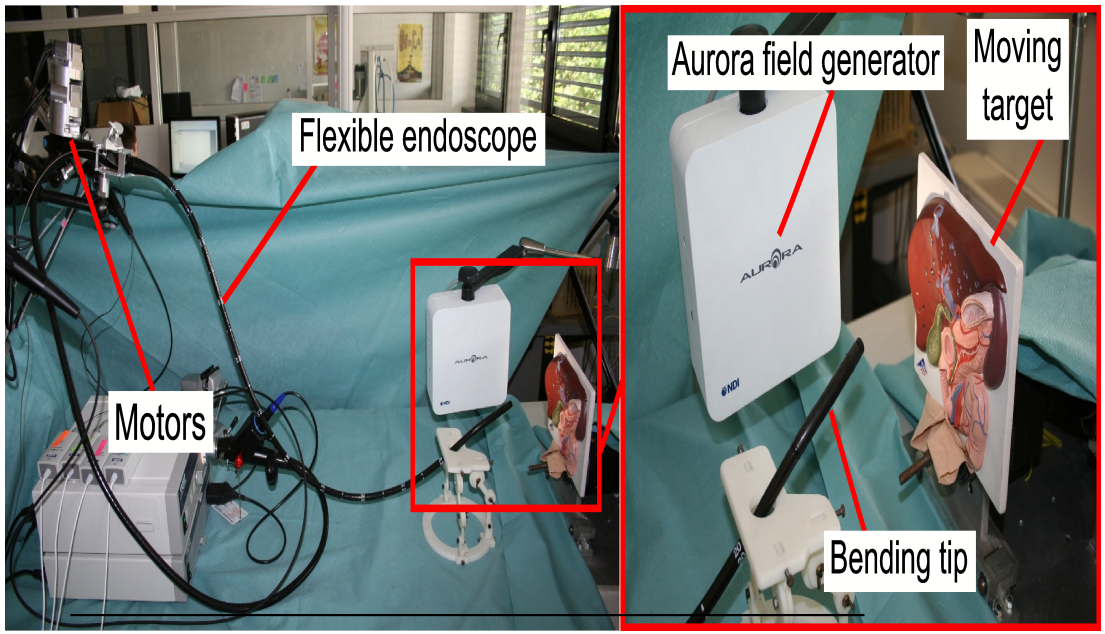
- Tracking électromagnétique & asservissement visuel[5]
  - Objets: Robot/Structure 3D
  - Roles: IHM/lecteurs/tracker/commande/visualisation

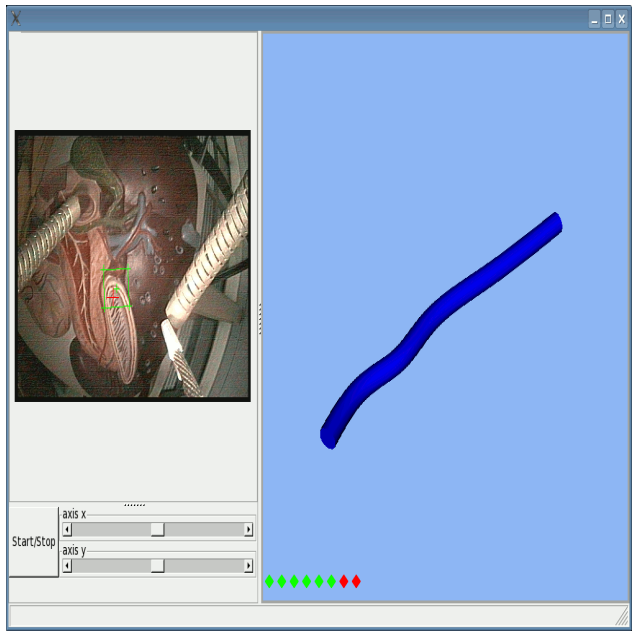


5. J.-B. FASQUEL et al. “A role-based component architecture for computer assisted interventions: illustration for electromagnetic tracking and robotized motion rejection in flexible endoscopy”. Dans : *Insight Journal, Workshop on Systems and Architecture for Computer Assisted Intervention, the 12th International Conference on Medical Image Computing and Computer Assisted Intervention, London*. 2009. URL : http://hdl.handle.net/10380/3069

Robotique

- Tracking électromagnétique & asservissement visuel[5]
  - Objets: Robot/Structure 3D
  - Roles: IHM/lecteurs/tracker/commande/visualisation





5. J.-B. FASQUEL et al. “A role-based component architecture for computer assisted interventions: illustration for electromagnetic tracking and robotized motion rejection in flexible endoscopy”. Dans : *Insight Journal, Workshop on Systems and Architecture for Computer Assisted Intervention, the 12th International Conference on Medical Image Computing and Computer Assisted Intervention, London*. 2009. URL : http://hdl.handle.net/10380/3069

1 Logiciels

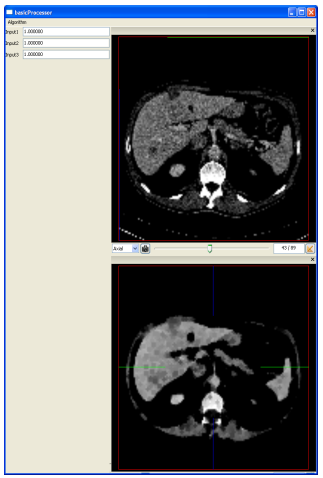
2 Algorithmes
- Contexte
- Contribution: régions d'intérêt (ROI)
- Illustration

3 Synthèse

Logiciel, Algorithmes & Programmation générique

Niveau logiciel

```
<object id="basicProcessor" type="Object">
 <object id="Input0" type="Image" .../>
 ...
 <service type="IProcessor" implementation="erode" >
   <Input id="Input0" .../>
   ...
 </service>
 ...
</object>
```



Niveau logiciel
Données dynamiques ("**Input0**")

```
class Image
{
...
void *buffer;
std::string pointType;
...
}
```

→

"Dispatching" → code générique

```
...
if( pointType == "Real" )
{
code< float >(...)
}
...
```

→

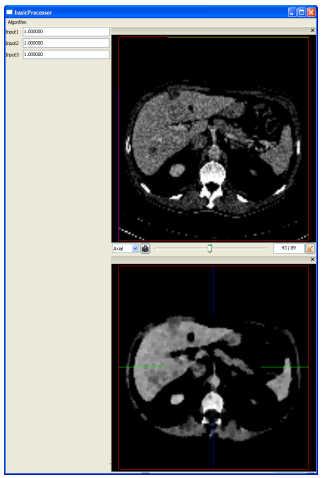
Niveau algorithme ("**erode**")
Code couramment utilisé

```
template
void code< TPOINT >(...)
{
...
}
...
```

Logiciel, Algorithmes & Programmation générique

Niveau logiciel

```
<object id="basicProcessor" type="Object">
 <object id="Input0" type="Image" .../>
 ...
 <service type="IProcessor" implementation="erode" >
   <Input id="Input0" .../>
   ...
 </service>
 ...
</object>
```



Niveau logiciel
Données dynamiques ("**Input0**")

```
class Image
{
...
void *buffer;
std::string pointType;
...
}
```

→

"Dispatching" → code générique

```
...
if( pointType == "Real" )
{
code< float >(...)
}
...
```

→

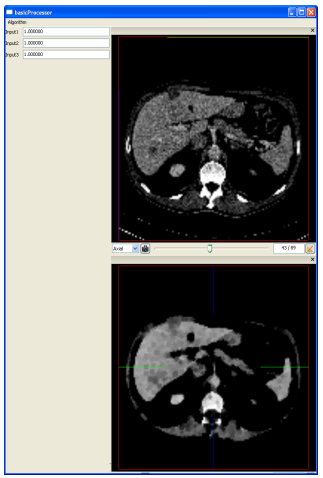
Niveau algorithme ("**erode**")
Code couramment utilisé

```
template
void code< TPOINT >(...)
{
...
}
...
```

Logiciel, Algorithmes & Programmation générique

Niveau logiciel

```
<object id="basicProcessor" type="Object">
 <object id="Input0" type="Image" .../>
 ...
 <service type="IProcessor" implementation="erode" >
   <Input id="Input0" .../>
   ...
 </service>
 ...
</object>
```



Niveau logiciel
Données dynamiques ("Input0")

```
class Image
{
...
void *buffer;
std::string pointType;
...
}
```

→

"Dispatching" → code générique

```
...
if( pointType == "Real" )
{
code< float >(...)
}
...
```

→

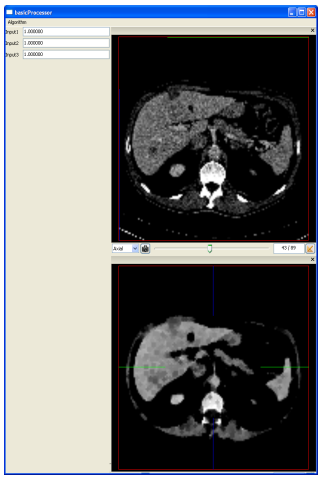
Niveau algorithme ("erode")
Code couramment utilisé

```
template
void code< TPOINT >(...)
{
...
}
...
```

Logiciel, Algorithmes & Programmation générique

Niveau logiciel

```
<object id="basicProcessor" type="Object">
 <object id="Input0" type="Image" .../>
 ...
 <service type="IProcessor" implementation="erode" >
   <Input id="Input0" .../>
   ...
 </service>
 ...
</object>
```



Niveau logiciel
Données dynamiques ("**Input0**")

```
class Image
{
...
void *buffer;
std::string pointType;
...
}
```

$\longrightarrow$

"Dispatching" $\longrightarrow$ code générique

```
...
if( pointType == "Real" )
{
code< float >(...)
}
...
```

$\longrightarrow$

Niveau algorithme ("**erode**")
Code couramment utilisé

```
template
void code< TPOINT >(...)
{
...
}
...
```

Environnement classique...

- Multiples frameworks génériques: ITK, VIGRA,...
- Code paramétrable au niveau des données (image)
  - Dimension spatiale (et temporelle):
    - 2D, 2D+t, 3D, 3D+t,...
  - Dimension photométrique
    - Scalaire, multicanaux (e.g. couleur), matrice
    - Dynamique et précision variable: N-bits, virgule flottante
- Code paramétrable au niveau de l'algorithme
  - Type des données d'entrées et de sorties

Exemple C++ avec ITK

Environnement classique...

- Multiples frameworks génériques: ITK, VIGRA,...
- Code paramétrable au niveau des données (image)
  - Dimension spatiale (et temporelle):
    - 2D, 2D+t, 3D, 3D+t,...
  - Dimension photométrique
    - Scalaire, multicanaux (e.g. couleur), matrice
    - Dynamique et précision variable: N-bits, virgule flottante
- Code paramétrable au niveau de l'algorithme
  - Type des données d'entrées et de sorties

Exemple C++ avec ITK

```
typedef itk::Image< float , 3 >   TImageIn;
typedef itk::Image< char , 3 >   TImageOut;
....
```

Environnement classique...

- Multiples frameworks génériques: ITK, VIGRA,...
- Code paramétrable au niveau des données (image)
  - Dimension spatiale (et temporelle):
    - 2D, 2D+t, 3D, 3D+t,...
  - Dimension photométrique
    - Scalaire, multicanaux (e.g. couleur), matrice
    - Dynamique et précision variable: N-bits, virgule flottante
- Code paramétrable au niveau de l'algorithme
  - Type des données d'entrées et de sorties

Exemple C++ avec ITK

```
typedef itk::Image< float , 3 >   TImageIn;
typedef itk::Image< char , 3 >   TImageOut;
....
itk::MyImageFilterType< TImageIn, TImageOut >::Pointer myFilter = ...::New();
```

Et la région traitée ?

- Réduction du temps de calcul
- Amélioration de l'efficacité

Et la région traitée ?

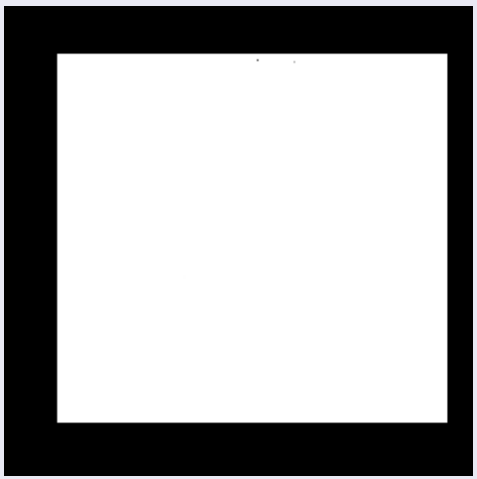
- Réduction du temps de calcul
- Amélioration de l'efficacité
- Les différentes "formes" d'une région: variabilité
  - Géométrique: carré, rectangle, ... cercle
  - Non géométrique: masque, liste de points

Et la région traitée ?

- Réduction du temps de calcul
- Amélioration de l'efficacité
- Les différentes "formes" d'une région: variabilité
  - Géométrique: carré, rectangle, ... cercle
  - Non géométrique: masque, liste de points

Image

Région d'intérêt: carré



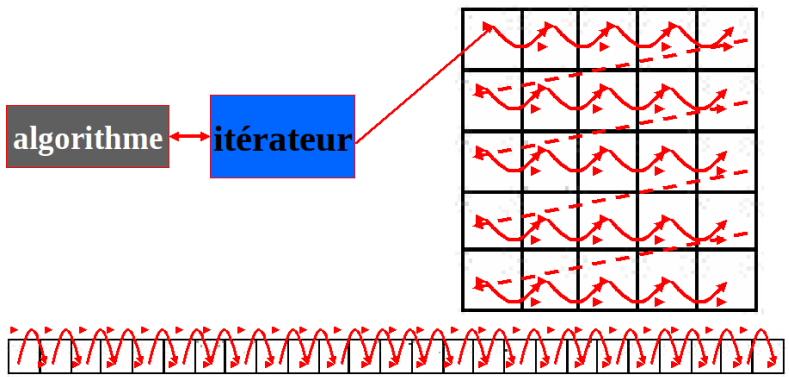
Région d'intérêt: masque

Gestion classique: régions rectangulaires

- Utilisation du patron de conception "Iterator"
  - Abstraction du parcours et accès aux données

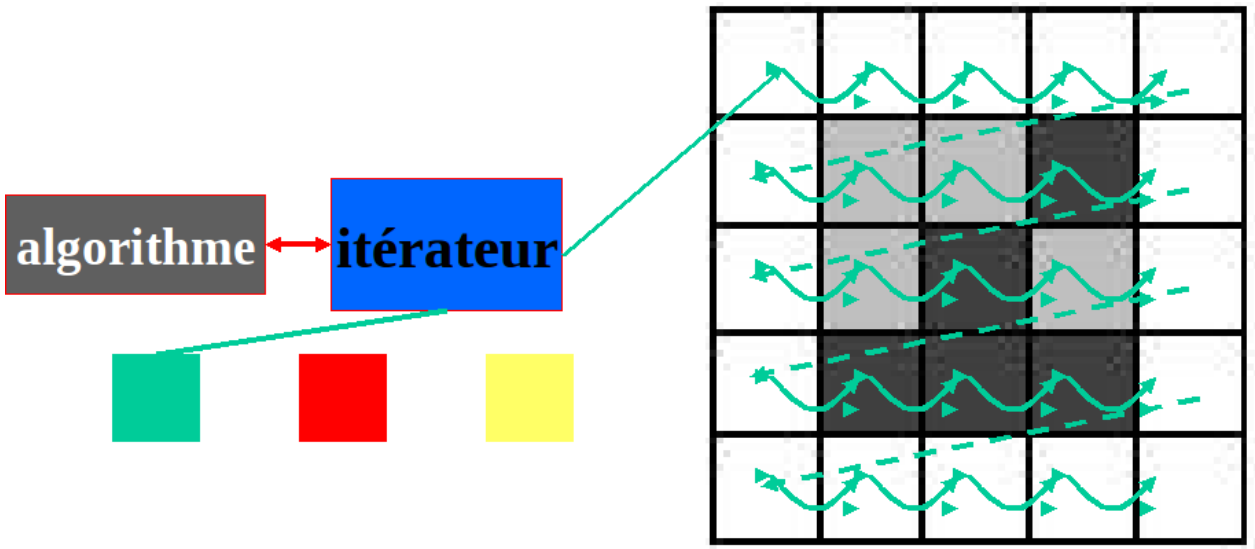
Algorithme et itérateur

```
void myAlgorithm(Iterator first, Iterator last, ...)
{
...
for(Iterator current=first;current!=last;++current)
{
 ...
 // Traitement
 // Accès par *current;
 ...
}
...
}
```



Un nouveau paramètre: le "scanner"

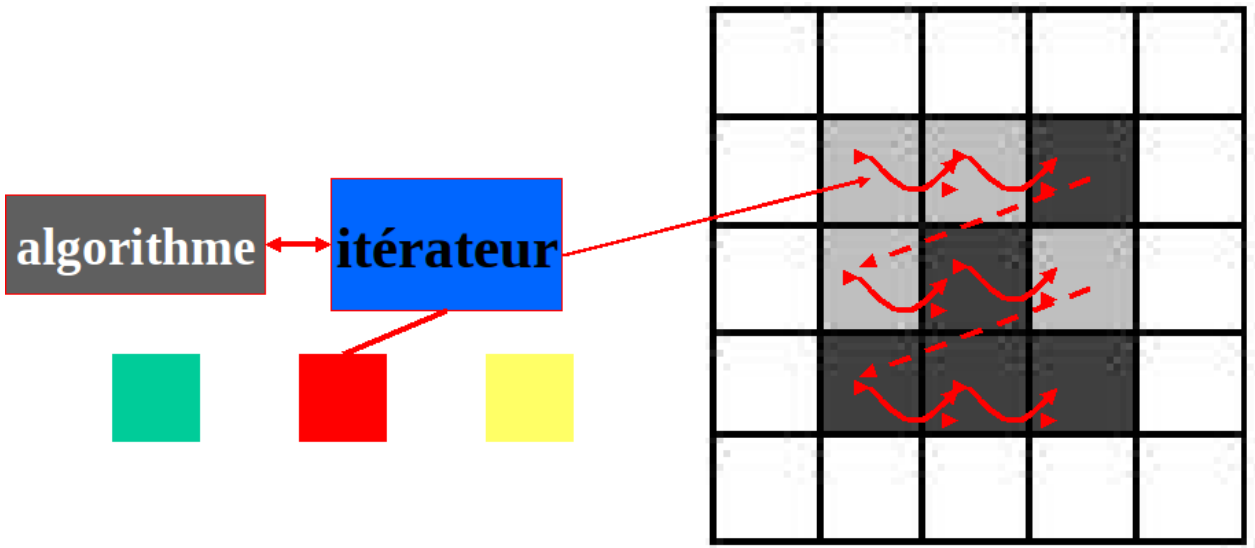
- Paramétrer l'itérateur par un "scanner" [6]



6. J.-B. FASQUEL, V. AGNUS et J. LAMY. "An efficient and generic extension to ITK to process arbitrary shaped regions of interest". Dans : *Computer Methods and Programs in Biomedicine* 81 (2006)

Un nouveau paramètre: le "scanner"

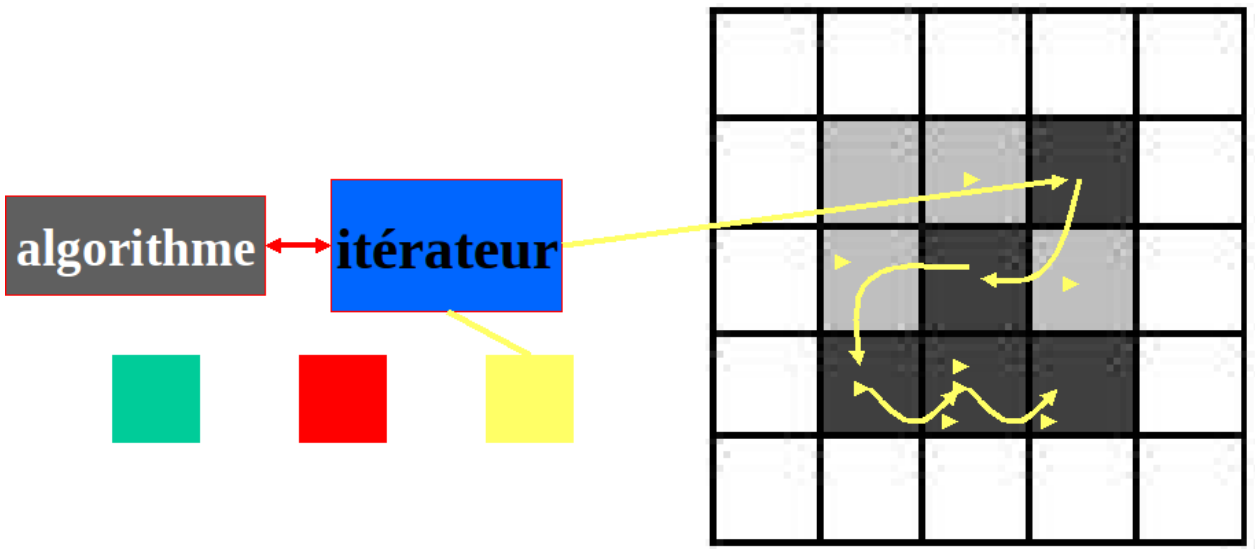
- Paramétrer l'itérateur par un "scanner" [6]



6. J.-B. FASQUEL, V. AGNUS et J. LAMY. "An efficient and generic extension to ITK to process arbitrary shaped regions of interest". Dans : *Computer Methods and Programs in Biomedicine* 81 (2006)

Un nouveau paramètre: le "scanner"

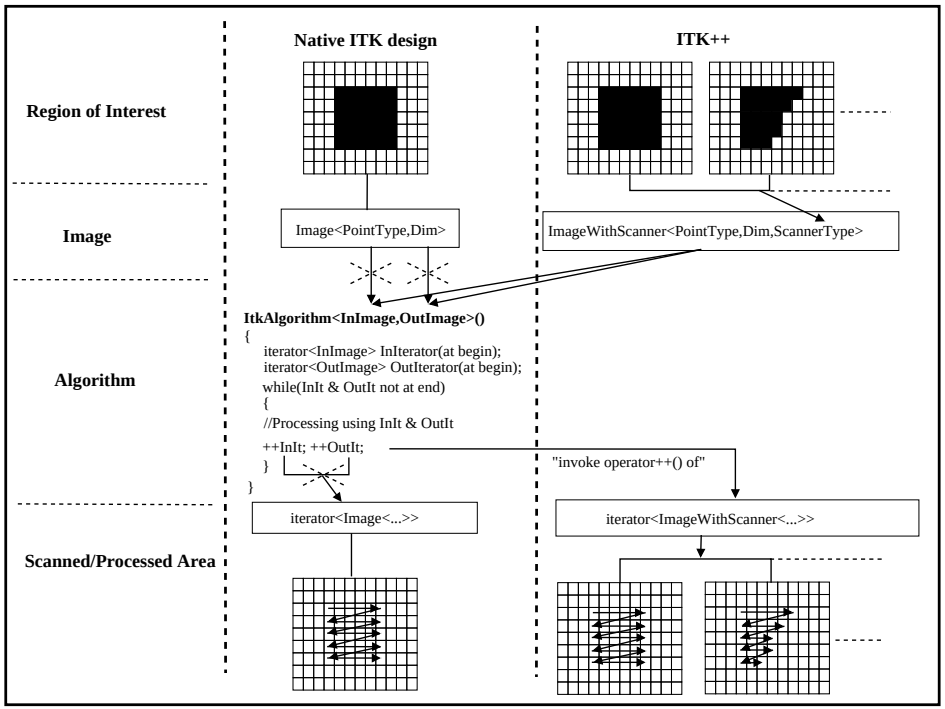
- Paramétrer l'itérateur par un "scanner" [6]



6. J.-B. FASQUEL, V. AGNUS et J. LAMY. "An efficient and generic extension to ITK to process arbitrary shaped regions of interest". Dans : *Computer Methods and Programs in Biomedicine* 81 (2006)

Implémentation dans ITK

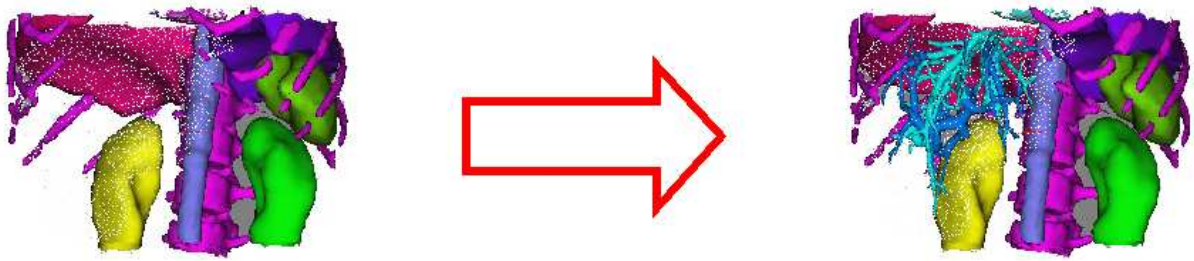
- `ImageWithScanner, ShapedScanner...`[7]

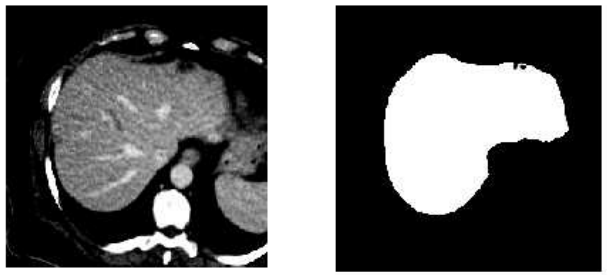


7. J.-B. FASQUEL, V. AGNUS et J. LAMY. "An efficient and generic extension to ITK to process arbitrary shaped regions of interest". Dans : *Computer Methods and Programs in Biomedicine* 81 (2006)

Simplification de la chaîne de traitement

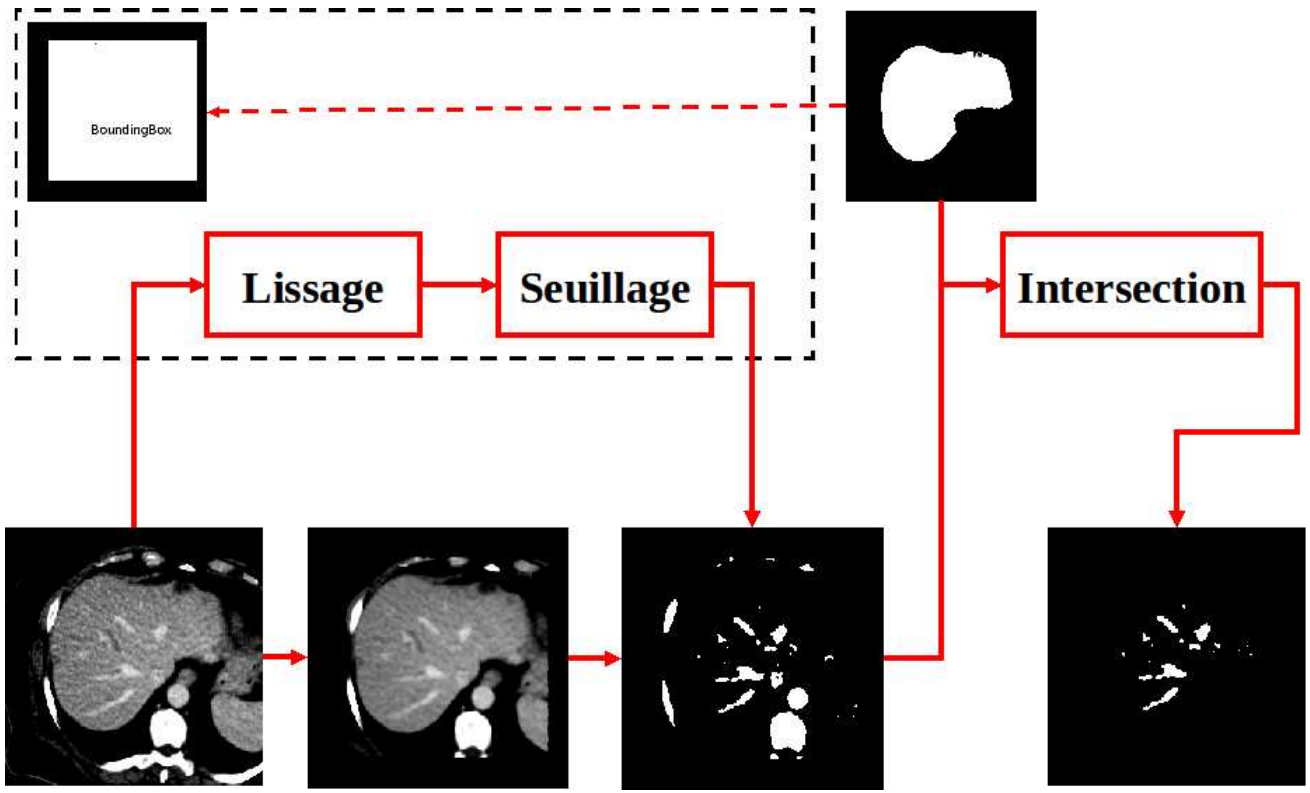
- Exemple de scénario





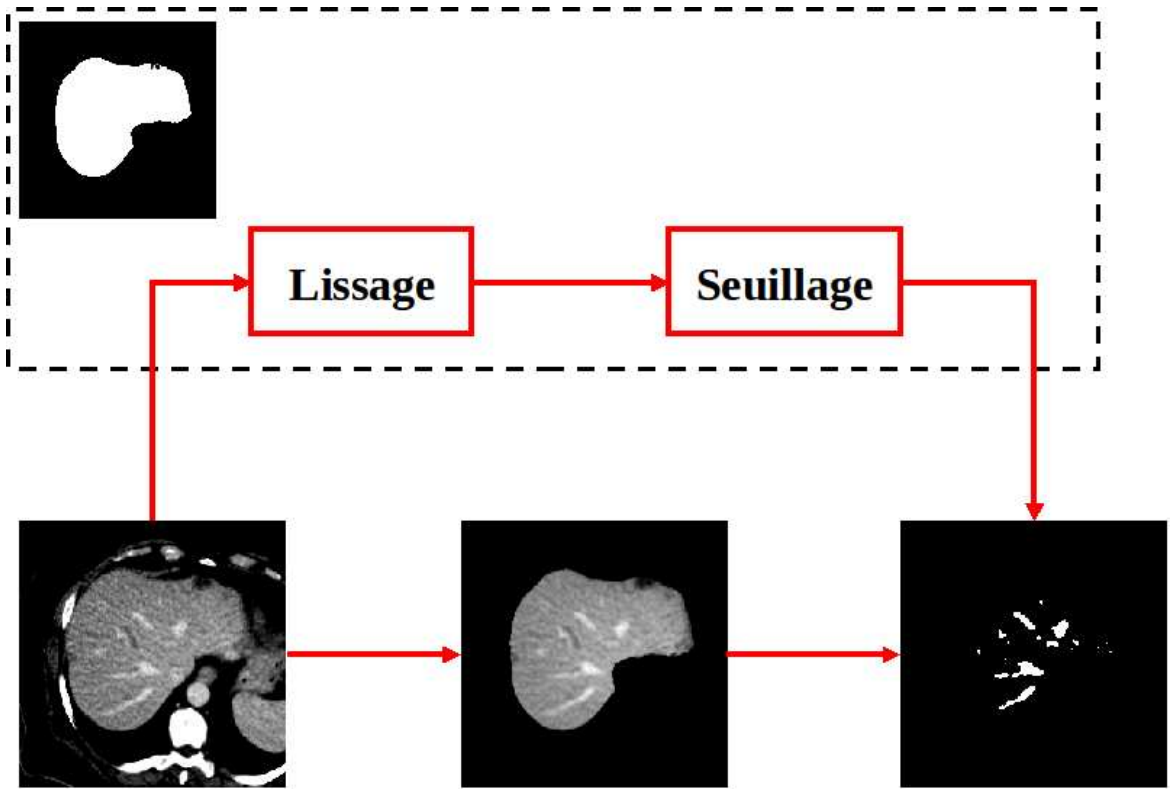
Simplification de la chaîne de traitement

- Chaîne de traitement native



Simplification de la chaîne de traitement

- Chaîne de traitement simplifiée



Simplification de la chaîne de traitement

Ecriture native
`TImage` ⟶ `itk::Image<>`

```
... //Creating bounding box from mask
// Defining ITK filters and setting parameters
itk::Filter1< TImage, TImage >::Pointer smoothing = ...::New();
smoothing->Set...; //Setting Parameters
itk::Filter2< TImage, TImage >::Pointer thresholding = ...::New();
thresholding->Set...; //Setting Parameters

// Pipeline: Connecting filters
smoothing->SetInput( reader->GetOutput());
thresholding->SetInput( smoothing->GetOutput() );
thresholding->GetOutput()->SetRequestedRegion( &BoundingBox);

// Executation
thresholding->Update();

// Getting result (smaller than the input)
TImage::Pointer croppedImage=thresholding->GetOutput();
croppedImage->DisconnectPipeline();

// Pasting the result in a larger image
itk::PasteImageFilter<TImage>::Pointer pasteFilter = ...::New();
pasteFilter->Set...; //Setting Parameters
pasteFilter->SetInput( croppedImage );

// Intersecting the initial liver mask with the enlarged result
itk::MaskImageFilter<...>::Pointer maskImageFilter=...::New();
maskImageFilter->SetInput( 1 , maskReader->GetOutput() );
maskImageFilter->SetInput( pasteFilter->GetOutput() );
maskImageFilter->Update();
...
```

Nouvelle écriture
`TImage` ⟶ `itk::ImageWithScanner<>`

```
... //Creating scanner from mask
// Defining ITK filters and setting parameters
itk::Filter1< TImage, TImage >::Pointer smoothing = ...::New();
smoothing->Set...; //Setting Parameters
itk::Filter2< TImage, TImage >::Pointer thresholding = ...::New();
thresholding->Set...; //Setting Parameters

// Pipeline: Connecting filters
smoothing->SetInput(reader->GetOutput());
thresholding->SetInput(smoothing->GetOutput());

// Setting scanner
thresholding->GetOutput()->SetScanner(&scanner);

// Executation
thresholding->Update();
...
```

Temps de traitement

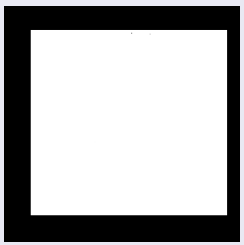
- Réduction significative[8 9]

Exemple: facteur $\simeq$ 3.5 (taille et vitesse)

Algorithme de lissage	filtrage anisotrope	dilatation morphologique	filtrage moyen
Parallélépipède optimal	52s	5.65s	6.66s
Masque	16s	1.62s	1.80s

Image

Parallélépipède (200 $\times$ 197 $\times$ 77)



Masque

8. J.-B. FASQUEL et al. "An Interactive Medical Segmentation System Based on the Optimal Management of Regions of Interest Using Topological Medical Knowledge". Dans : *Computer Methods and Programs in Biomedicine* 82 (2006), p. 216–230

9. J.-B. FASQUEL, V. AGNUS et J. LAMY. "An efficient and generic extension to ITK to process arbitrary shaped regions of interest". Dans : *Computer Methods and Programs in Biomedicine* 81 (2006)

Qualité de traitement

- Topologie, segmentation hiérarchique et ROI [10]
- Exemple: $ROI_{foie} \subseteq ROI_{peau} \setminus \{ROI_{rate} \bigcup ROI_{poumons} \ldots\}$

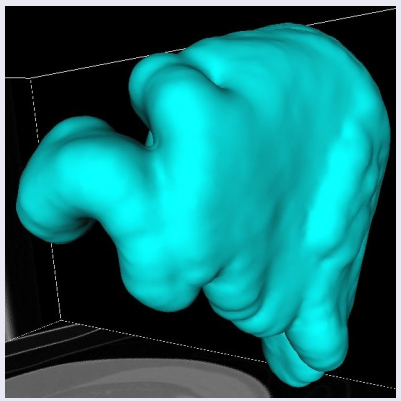
Exemple segmentation rate puis foie

10. J.-B. FASQUEL et al. "An Interactive Medical Segmentation System Based on the Optimal Management of Regions of Interest Using Topological Medical Knowledge". Dans : *Computer Methods and Programs in Biomedicine* 82 (2006), p. 216–230

Qualité de traitement

- Topologie, segmentation hiérarchique et ROI [10]
- Exemple: $ROI_{foie} \subseteq ROI_{peau} \setminus \{ROI_{rate} \bigcup ROI_{poumons} \ldots\}$

Exemple segmentation rate puis foie
"SANS ROI" → le foie englobe la rate

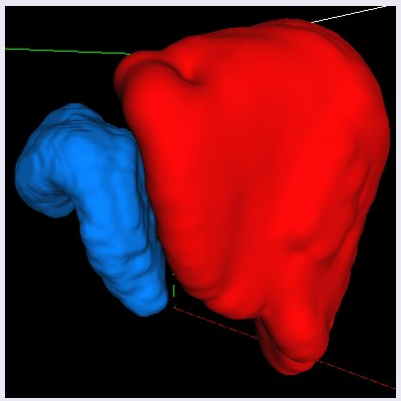


10. J.-B. FASQUEL et al. "An Interactive Medical Segmentation System Based on the Optimal Management of Regions of Interest Using Topological Medical Knowledge". Dans : *Computer Methods and Programs in Biomedicine* 82 (2006), p. 216–230

Qualité de traitement

- Topologie, segmentation hiérarchique et ROI [10]
- Exemple: $ROI_{foie} \subseteq ROI_{peau} \setminus \{ROI_{rate} \bigcup ROI_{poumons} \ldots\}$

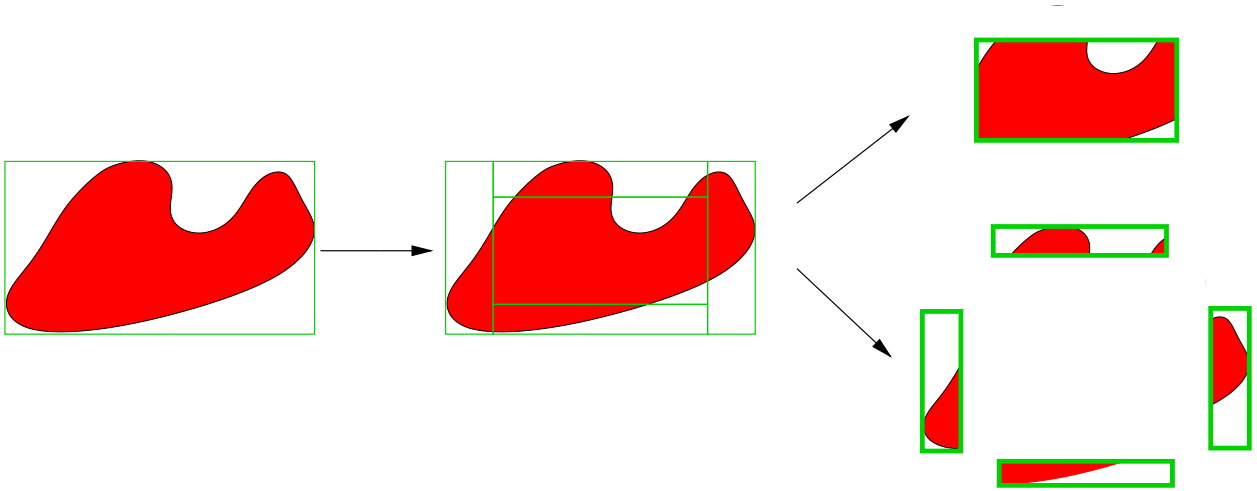
Exemple segmentation rate puis foie
"AVEC ROI" $\longrightarrow$ foie & rate distincts



10. J.-B. FASQUEL et al. "An Interactive Medical Segmentation System Based on the Optimal Management of Regions of Interest Using Topological Medical Knowledge". Dans : *Computer Methods and Programs in Biomedicine* 82 (2006), p. 216–230

Difficultés

- Occupation mémoire
  - Masque plus consommateur qu'un simple rectangle
  - Liste de points ou masque ?
- Traitement par blocs (e.g. multithreading)
  - Gestion du découpage optimal



Difficultés

- Extrapolation
  - Cas régions convexes (rectangle): "simple"
    - e.g. filtrage linéaire : recopie de la bordure
    - e.g. dilation morphologique : valeur $-\infty$
  - Cas régions non convexes: valeurs aux concavités ?
    - Prise en compte du voisinage: calculs supplémentaires

5	5	5	4	5	6	6	6	?
5	5	5	4	5	6	6	?	8
5	5	5	4	5	6	?	8	8
4	?	3	4	6	7	8	8	8
4	4	5	?	?	7	7	7	7
2	2	3	3	7	?	9	9	9
2	?	1	1	?	8	8	8	8
?	1	1	1	?	8	8	8	8
1	1	1	1	?	8	8	8	8

1 Logiciels

2 Algorithmes

3 Synthèse

Bilan

- Roles & composants
  - Approche prometteuse
  - Difficultés
    - Utilisation optimale pour algorithme/visualisation/IHM/... ?
    - Configurabilité & complexité: quel équilibre ?
    - Dépendances inter-roles/composants, modélisation, ...

Niveau logiciel

```
<object id="basicProcessor" type="Object">
 <object id=" Input0" type="Image" >
   <service type="IRender" implementation="myRender" .../>
   <service type="IReader" implementation="myReader" .../>
 </object>
 ...
 <service type="IProcessor" implementation=" erode" >
   <Input id=" Input0" .../>
   ...
 </service>
 <service type="IAspect" implementation="myAspect" .../>
</object>
```

→

"Dispatching" ⟶ code générique

```
...
if( pointType == "Real" )
{
code< float >(...)
}
...
```

→

Niveau algorithme ("erode")

```
...
template
void code< TPOINT >(...)
{
...
}
...
```

Bilan

- Programmation générique en traitement d'images
  - Dimension spatiale/temporelle/photométrique
  - Type entrée(s)/sortie(s) d'un algorithme
  - Régions d'intérêt
    - Codage/Temps de calcul/Qualité
    - Difficultés : blocs/extrapolation/mémoire

Niveau logiciel

```
<object id="basicProcessor" type="Object">
 <object id=" Input0" type="Image" >
   <service type="IRender" implementation="myRender" .../>
   <service type="IReader" implementation="myReader" .../>
 </object>
 ...
 <service type="IProcessor" implementation=" erode" >
   <Input id=" Input0" .../>
   ...
 </service>
 <service type="IAspect" implementation="myAspect" .../>
</object>
```

→

"Dispatching" ⟶ code générique

```
...
if( pointType == "Real" )
{
code< float >(...)
}
...
```

→

Niveau algorithme ("erode")

```
...
template
void code< TPOINT >(...)
{
...
}
...
```

Bilan

- Programmation générique en traitement d'images
  - Dimension spatiale/temporelle/photométrique
  - Type entrée(s)/sortie(s) d'un algorithme
  - Régions d'intérêt
    - Codage/Temps de calcul/Qualité
    - Difficultés : blocs/extrapolation/mémoire

Niveau logiciel

```
<object id="basicProcessor" type="Object">
 <object id=" Input0" type="Image" >
   <service type="IRender" implementation="myRender" .../>
   <service type="IReader" implementation="myReader" .../>
 </object>
 ...
 <service type="IProcessor" implementation=" erode" >
   <Input id=" Input0" .../>
   ...
 </service>
 <service type="IAspect" implementation="myAspect" .../>
</object>
```

→

"Dispatching" ⟶ code générique

```
...
if( pointType == "Real" )
{
code< float >(...)
}
...
```

→

Niveau algorithme ("erode")

```
...
template
void code< TPOINT >(...)
{
...
}
...
```

Quelques références bibliographiques...

- Logiciel [11] [12] [13]
- Régions d'intérêt [14] [15]

---

11. J.-B. FASQUEL et al. "A modular and evolutive component oriented software architecture for patient modeling". Dans : *Computer Methods and Programs in Biomedicine* 83 (2006), p. 222–233

12. J.-B. FASQUEL et al. "A XML based component oriented architecture for image guided surgery: illustration for the video based tracking of a surgical tool". Dans : *Insight Journal, Workshop on Systems and Architectures for Computer Assisted Interventions, 11th International Conference on Medical Image Computing and Computer Assisted Intervention*. 2008. URL : http://hdl.handle.net/10380/1497

13. J.-B. FASQUEL et al. "A role-based component architecture for computer assisted interventions: illustration for electromagnetic tracking and robotized motion rejection in flexible endoscopy". Dans : *Insight Journal, Workshop on Systems and Architecture for Computer Assisted Intervention, the 12th International Conference on Medical Image Computing and Computer Assisted Intervention, London*. 2009. URL : http://hdl.handle.net/10380/3069

14. J.-B. FASQUEL, V. AGNUS et J. LAMY. "An efficient and generic extension to ITK to process arbitrary shaped regions of interest". Dans : *Computer Methods and Programs in Biomedicine* 81 (2006)

15. J.-B. FASQUEL et al. "An Interactive Medical Segmentation System Based on the Optimal Management of Regions of Interest Using Topological Medical Knowledge". Dans : *Computer Methods and Programs in Biomedicine* 82 (2006), p. 216–230

Disponibilité du code

- Aspect logiciel
  - Ouverture Open-Source en Septembre 2009 par l'IRCAD
  - Fw4spl: FrameWork for Software Product Line
  - http://code.google.com/p/fw4spl/
  - Multiplateforme Linux32-64/Win32-64/Osx
  - Noyau complet
  - Fonctionnalités limitées: IHM, lecture, visualisation
  - Tutoriaux/Contributeurs/Publications
  - Package externes: vtk, wxwidgets, boost,...
  - Projet Open source complémentaire: Sconspiracy
    - Scripts pour la compilation C++ multiplateforme
    - Basé sur SCons (Python)
    - http://code.google.com/p/sconspiracy/
- Aspect Algorithme: ouverture opensource prévue...