Morph-M et généricité en traitement d'images.

Séminaire LRDE

Raffi Enficiaud
DxO Labs

Romain Lerallut
A2IA

30 janvier 2008

(Ex.) Centre de Morphologie Mathématique - École des Mines de Paris

Généricité

Intérêt de la généricité en traitement d'images

- Pourquoi la généricité ?
- Quels problèmes peut-elle résoudre ?
- Ce qu'elle permet de faire
- Ce qu'elle ne fait pas (ou pas très bien)
- Ce qu'elle ne permet pas de faire

Traitements et fonctions

Problème $n^o 1$: duplication fonctionnelle

Exemple arithmétique : addition d'une constante « v » sur une image.

$\mathcal{F} : \forall p \in \mathbf{E}\ ,\ \mathcal{J}\ (p) = \mathcal{I}\ (p) + v$

Nombre de permutations

Traitements et fonctions

Problème n^o1 : duplication fonctionnelle

Exemple arithmétique : addition d'une constante « v » sur une image.

$$\mathcal{F} : \forall p \in \mathbf{E} \ , \ \mathcal{J}\ (p) = \mathcal{I}\ (p) + v$$

Dimension de **E**	Représentation de $\mathcal{I}$	Représentation de v	Représentation de $\mathcal{J}$
2D			

Nombre de permutations

Traitements et fonctions

Problème $n^o 1$: duplication fonctionnelle

Exemple arithmétique : addition d'une constante « v » sur une image.

$$\mathcal{F} : \forall p \in \mathbf{E} \ , \ \mathcal{J}\ (p) = \mathcal{I}\ (p) + v$$

Dimension de **E**	Représentation de $\mathcal{I}$	Représentation de v	Représentation de $\mathcal{J}$
2D			

Nombre de permutations

Traitements et fonctions

Problème $n^o 1$: duplication fonctionnelle

Exemple arithmétique : addition d'une constante « v » sur une image.

$$\mathcal{F} : \forall p \in \mathbf{E} \ , \ \mathcal{J}\ (p) = \mathcal{I}\ (p) + v$$

Dimension de **E**	Représentation de $\mathcal{I}$	Représentation de v	Représentation de $\mathcal{J}$
2D	**scalaire**	**scalaire**	
	entier « 8 bits »	**entier « 8 bits »**	

Nombre de permutations

Traitements et fonctions

Problème $n^o 1$: duplication fonctionnelle

Exemple arithmétique : addition d'une constante « v » sur une image.

$$\mathcal{F} : \forall p \in \mathbf{E} \ , \ \mathcal{J}\ (p) = \mathcal{I}\ (p) + v$$

Dimension de **E**	Représentation de $\mathcal{I}$	Représentation de v		Représentation de $\mathcal{J}$
2D	**scalaire**	**scalaire**	. . . . .	
	entier « 8 bits »	**entier « 8 bits »**		

Nombre de permutations

Traitements et fonctions

Problème $n^o 1$: duplication fonctionnelle

Exemple arithmétique : addition d'une constante « v » sur une image.

$$\mathcal{F} : \forall p \in \mathbf{E} \ , \ \mathcal{J}\ (p) = \mathcal{I}\ (p) + v$$

Dimension de **E**	Représentation de $\mathcal{I}$	Représentation de v		Représentation de $\mathcal{J}$
2D	**scalaire**	**scalaire**	. . . . .	**scalaire**
	entier « 8 bits »	**entier « 8 bits »**		**entier « 16 bits »**

Nombre de permutations

Traitements et fonctions

Problème $n^o 1$: duplication fonctionnelle

Exemple arithmétique : addition d'une constante « v » sur une image.

$$\mathcal{F} : \forall p \in \mathbf{E} \ , \ \mathcal{J}\ (p) = \mathcal{I}\ (p) + v$$

Dimension de **E**	Représentation de $\mathcal{I}$	Représentation de v		Représentation de $\mathcal{J}$
2D	**scalaire**	**scalaire**	. . . . .	**scalaire**
	entier « 8 bits »	**entier « 8 bits »**		**entier « 16 bits »**

Nombre de permutations

nb dimensions

2

Traitements et fonctions

Problème $n^o 1$: duplication fonctionnelle

Exemple arithmétique : addition d'une constante « v » sur une image.

$$\mathcal{F} : \forall p \in \mathbf{E} \ , \ \mathcal{J}\ (p) = \mathcal{I}\ (p) + v$$

Dimension de **E**	Représentation de $\mathcal{I}$	Représentation de v		Représentation de $\mathcal{J}$
2D	**scalaire**	**scalaire**	. . . . .	**scalaire**
	entier « 8 bits »	**entier « 8 bits »**		**entier « 16 bits »**

Nombre de permutations

$$\text{nb dimensions} \times \text{nb types}$$
$$2 \times 4$$

Traitements et fonctions

Problème $n^o 1$: duplication fonctionnelle

Exemple arithmétique : addition d'une constante « v » sur une image.

$$\mathcal{F} : \forall p \in \mathbf{E} \ , \ \mathcal{J}\ (p) = \mathcal{I}\ (p) + v$$

Dimension de **E**	Représentation de $\mathcal{I}$	Représentation de v		Représentation de $\mathcal{J}$
2D	**scalaire**	**scalaire**	. . . . .	**scalaire**
	entier « 8 bits »	**entier « 8 bits »**		**entier « 16 bits »**

Nombre de permutations

$$\text{nb dimensions} \times \text{nb types} \times \text{nb types}$$
$$2 \times 4 \times 4$$

Traitements et fonctions

Problème $n^o 1$: duplication fonctionnelle

Exemple arithmétique : addition d'une constante « v » sur une image.

$$\mathcal{F} : \forall p \in \mathbf{E} \ , \ \mathcal{J}\ (p) = \mathcal{I}\ (p) + v$$

Dimension de **E**	Représentation de $\mathcal{I}$	Représentation de v		Représentation de $\mathcal{J}$
2D	**scalaire**	**scalaire**	. . . . .	**scalaire**
	entier « 8 bits »	**entier « 8 bits »**		**entier « 16 bits »**

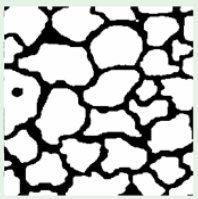
Nombre de permutations

$$(\text{nb dimensions} \times \text{nb types})^2 \times \text{nb types}$$
$$(2 \times 4)^2 \times 4 = 256$$

Traitements

Problème $n^o 2$: redondance algorithmique

Labélisation

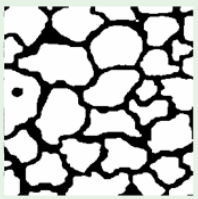


Mesures du nombre
de composantes

Traitements

Problème $n^o 2$: redondance algorithmique

Labélisation



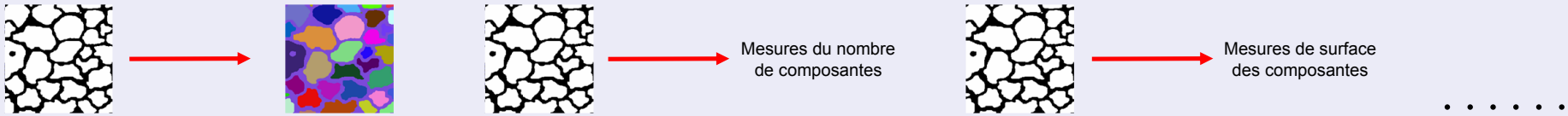
Mesures du nombre de composantes

Développements trop spécifiques à une application !

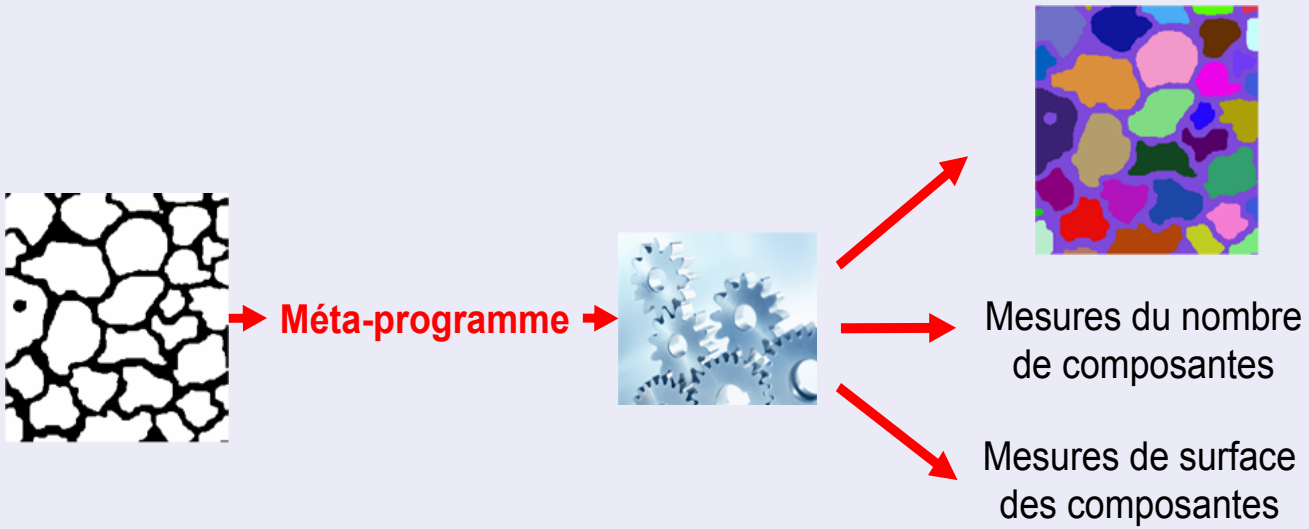
Traitements

Problème $n^o 2$: redondance algorithmique

Sans méta-programmation



Avec méta-programmation



Généricité par la Méta-programmation

Objectifs

Objectifs de la méta-programmation

1. Concentration des efforts sur le développement des algorithmes
2. Capitalisation
3. Réutilisation efficace de l'existant
4. Portage algorithmique

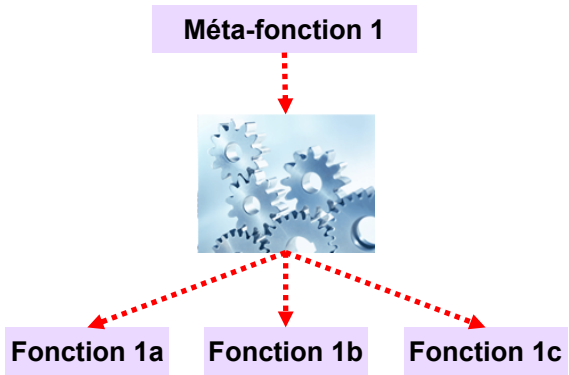
Méta-programmation

Définition

Méta-programmation ?

Résolution de nombreuses tâches par le compilateur

1. Résolution des types



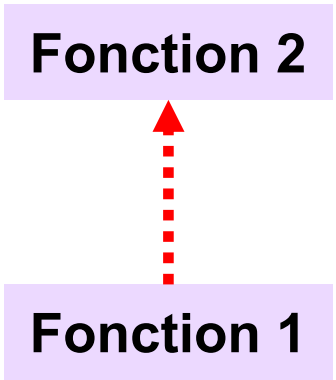
Méta-programmation

Définition

Méta-programmation ?

Résolution de nombreuses tâches par le compilateur

1. Résolution des types
2. Spécialisation



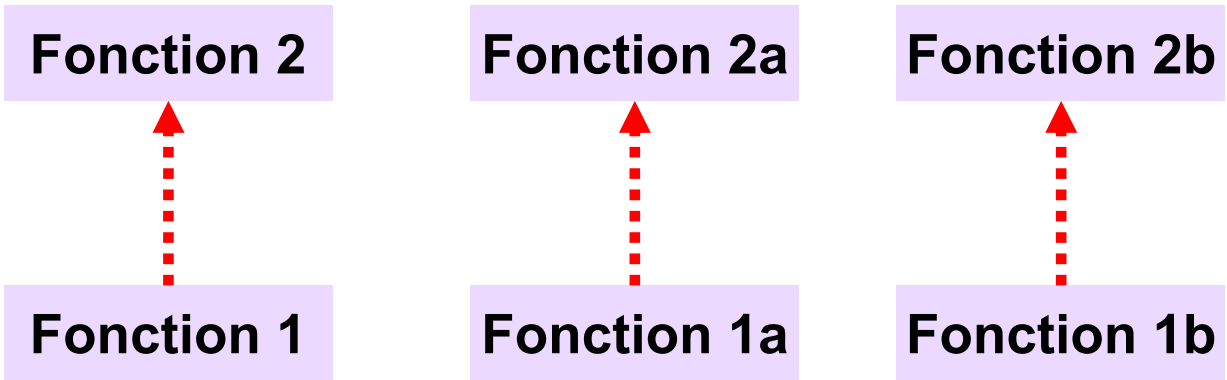
Méta-programmation

Définition

Méta-programmation ?

Résolution de nombreuses tâches par le compilateur

1. Résolution des types
2. Spécialisation



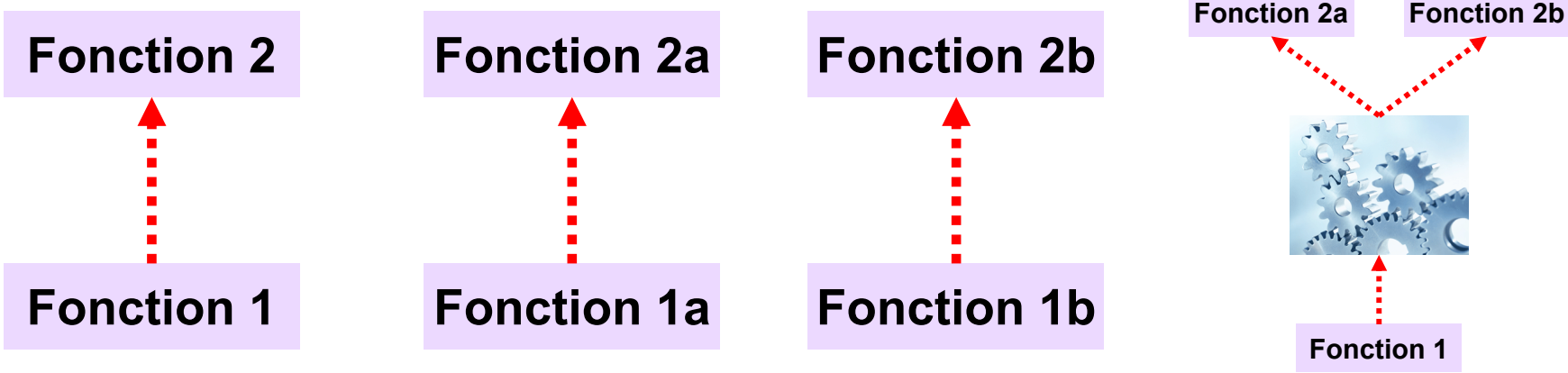
Méta-programmation

Définition

Méta-programmation ?

Résolution de nombreuses tâches par le compilateur

1. Résolution des types
2. Spécialisation



Morph-M et généricité en traitement d'images.

Part 1 : Généricité & traitement d'images

Plan

Généricité & traitement d'images

Algorithmes

Définition

Definition (Algorithme (Larousse))

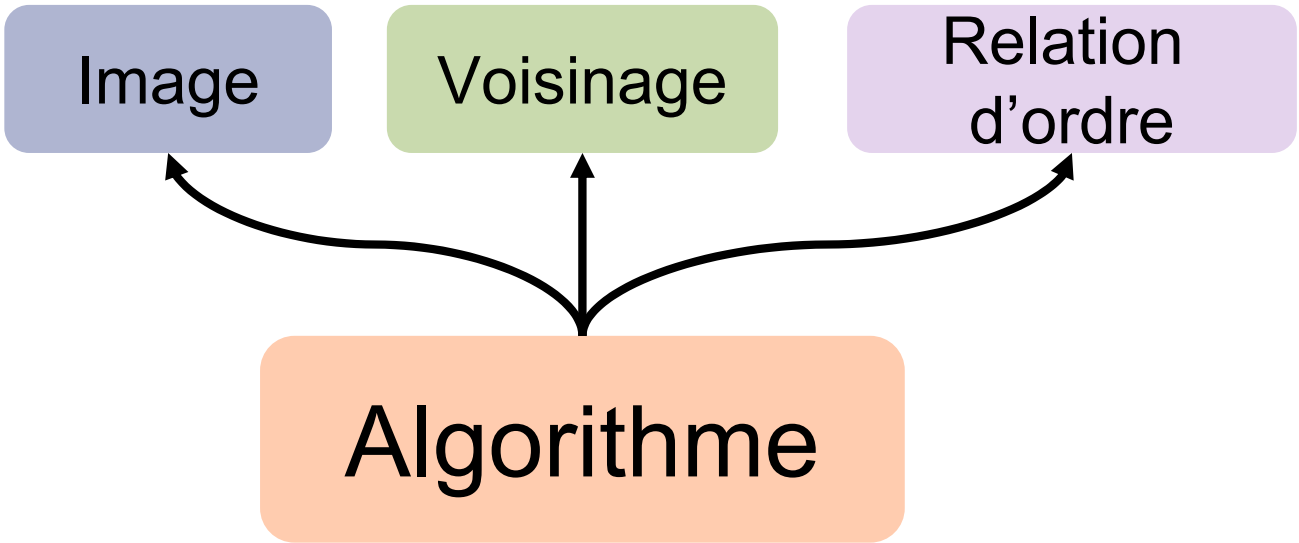
Suite finie d'opérations élémentaires constituant un schéma de calcul ou de résolution d'un problème.

Algorithmes

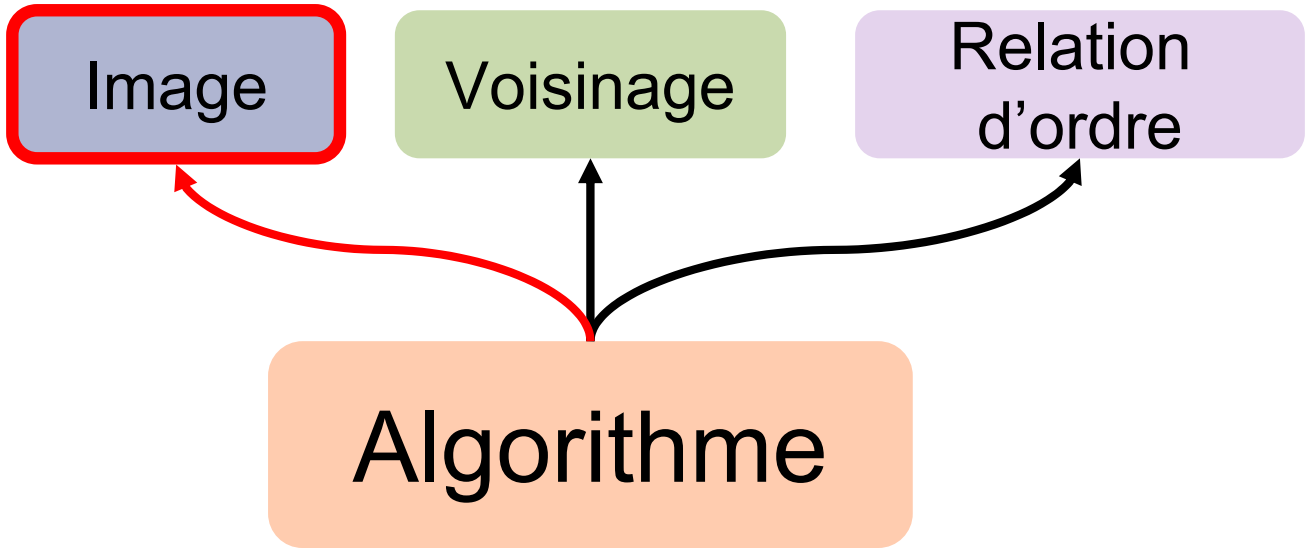
Définition

Definition (Algorithme (Larousse))

Suite finie d'opérations élémentaires constituant un schéma de calcul ou de résolution d'un problème.



Algorithmes & Images



Images

Définition

Definition

Une image est une application $\mathcal{I} : \mathbf{E} \longrightarrow \mathbf{F}$

- $\mathbf{E}$: espace des coordonnées
- $\mathbf{F}$: espace des valeurs

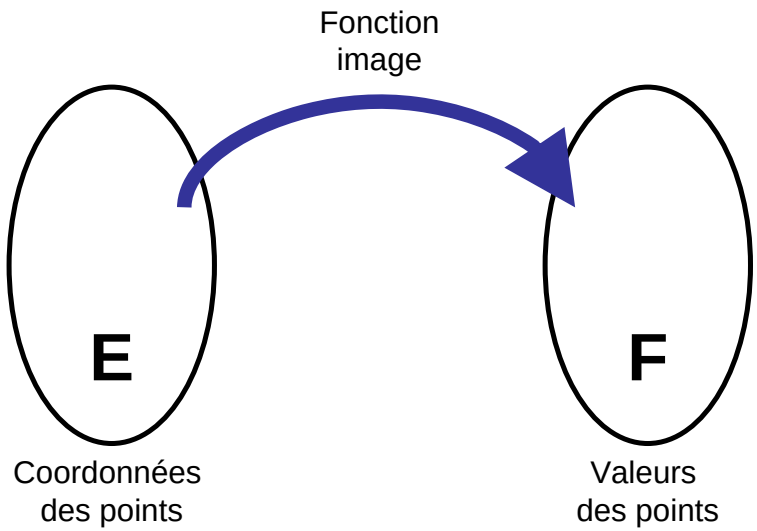
Images

Définition

Definition

Une image est une application $\mathcal{I} : \mathbf{E} \longrightarrow \mathbf{F}$

- **E** : espace des coordonnées
- **F** : espace des valeurs



Parcours des ensembles par itérateur

Multidimensionnalité d'un algorithme

Écrire « $\forall p \in \mathbf{E} \ldots$ »

Approche « classique »

Récupération des dimensions sur chaque axe

Parcours des ensembles par itérateur

Multidimensionnalité d'un algorithme

Écrire « $\forall p \in \mathbf{E} \ldots$ »

Approche « classique »

Récupération des dimensions sur chaque axe

Pour z de Z_{min} à Z_{max} {
    Pour y de Y_{min} à Y_{max} {
        Pour x de X_{min} à X_{max} {

}}}

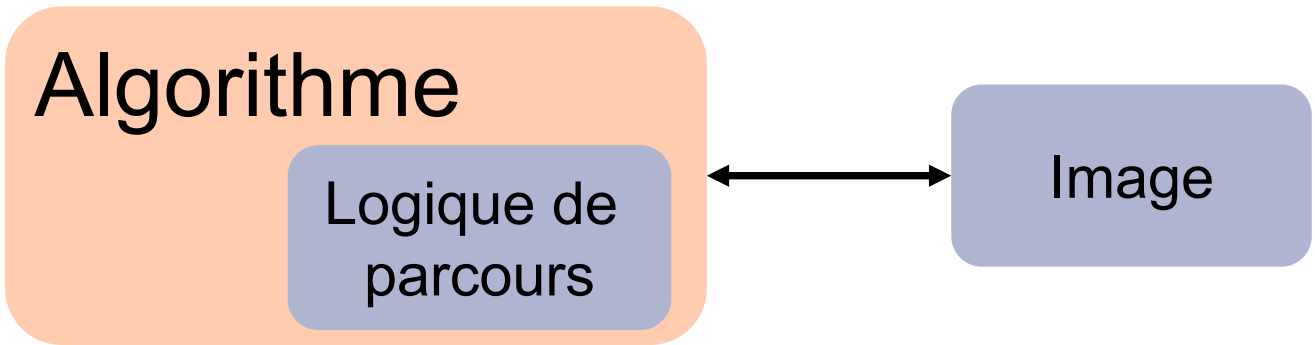
Parcours des ensembles par itérateur

Multidimensionnalité d'un algorithme

Écrire « $\forall p \in \mathbf{E} \ldots$ »

Approche « classique »

Récupération des dimensions sur chaque axe
Pour z de Z_{min} à Z_{max} {
    Pour y de Y_{min} à Y_{max} {
        Pour x de X_{min} à X_{max} {
            Traitement au point (x, y, z)
}}}



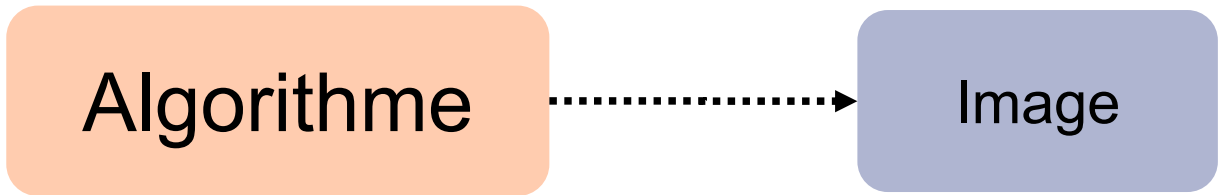
Parcours des ensembles par itérateur

Multidimensionnalité d'un algorithme

Écrire « $\forall p \in \mathbf{E} \ldots$ »

Approche par « itérateurs »

Récupération d'un « objet » de parcours de l'image : *it*



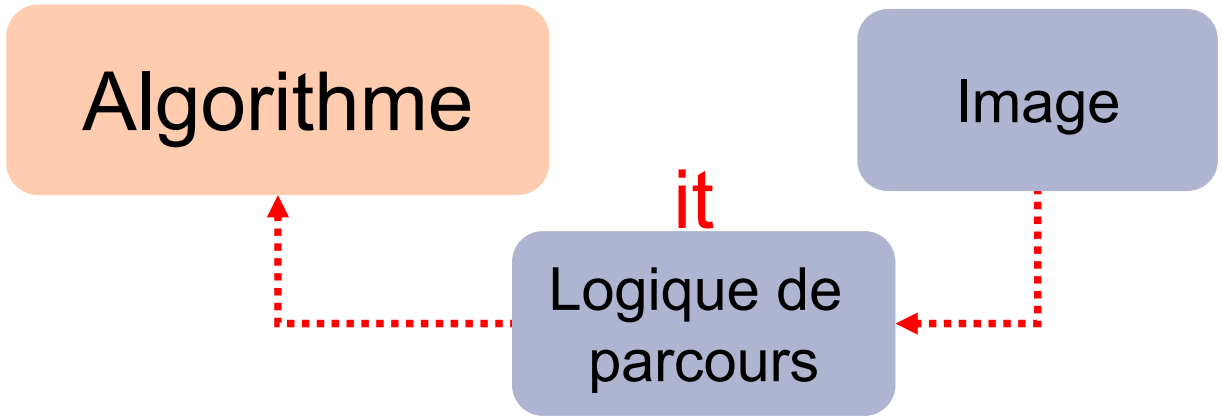
Parcours des ensembles par itérateur

Multidimensionnalité d'un algorithme

Écrire « $\forall p \in \mathbf{E} \ldots$ »

Approche par « itérateurs »

Récupération d'un « objet » de parcours de l'image : *it*



Parcours des ensembles par itérateur

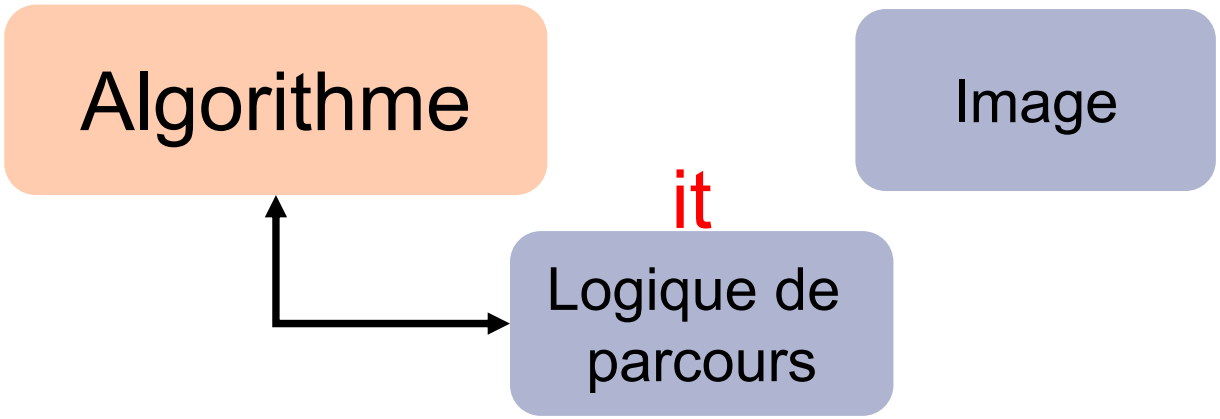
Multidimensionnalité d'un algorithme

Écrire « $\forall p \in \mathbf{E} \ldots$ »

Approche par « itérateurs »

Récupération d'un « objet » de parcours de l'image : *it*
Tant que *it* n'indique pas la fin du parcours {

}



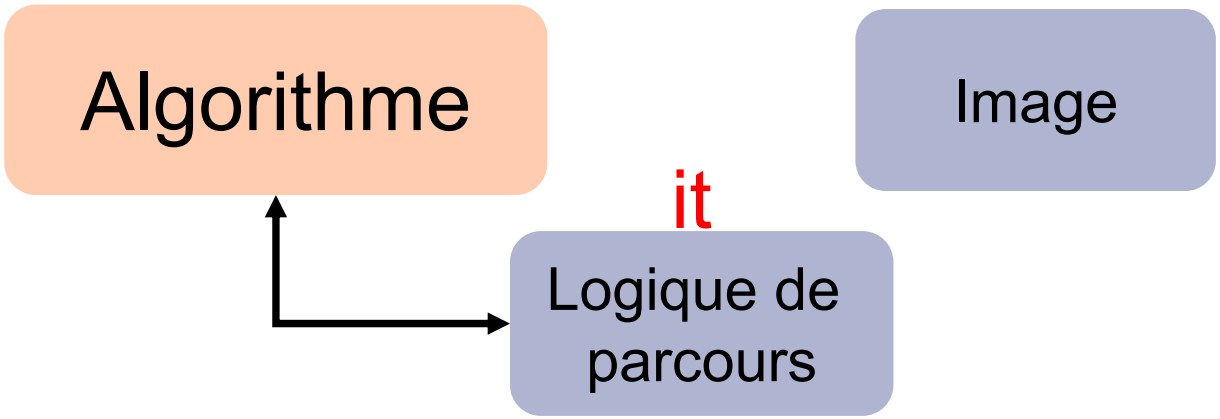
Parcours des ensembles par itérateur

Multidimensionnalité d'un algorithme

Écrire « $\forall p \in \mathbf{E} \ldots$ »

Approche par « itérateurs »

Récupération d'un « objet » de parcours de l'image : *it*
Tant que *it* n'indique pas la fin du parcours {
    Traitement point *p* fourni par *it*
}



Parcours des images par itérateur

Multidimensionnalité d'un algorithme

Itérateurs

Méthode universelle de parcours d'ensemble

Conséquence

1. Les structures proposent une méthode de parcours de leur support ou d'un sous-ensemble.

Parcours des images par itérateur

Multidimensionnalité d'un algorithme

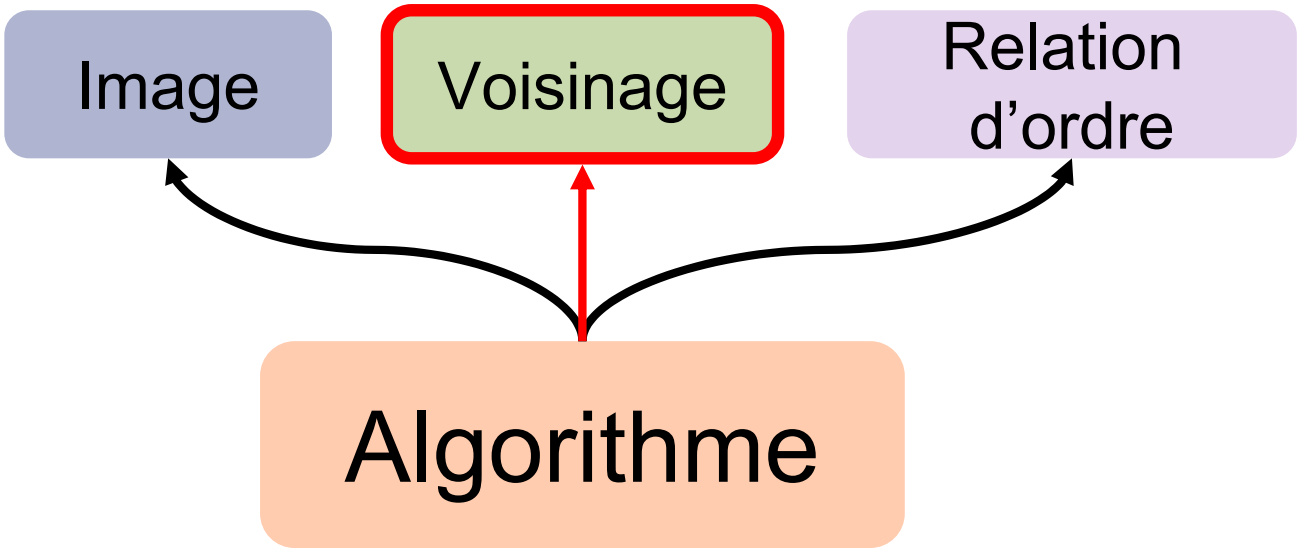
Itérateurs

Méthode universelle de parcours d'ensemble

Conséquence

1. Les structures proposent une méthode de parcours de leur support ou d'un sous-ensemble.
2. Les algorithmes deviennent indépendants
   1. du système de coordonnées des images (2D, 3D, 4D, ...).
   2. de la géométrie des images (taille, comportement sur les bords, ...)

Algorithmes & Voisinages



Voisinages

Définition

Definition

Un voisinage centré en $\mathbf{x}$ est un sous-ensemble $\mathcal{V}_{\mathbf{x}} \subset \mathbf{E}$, déterminé par une fonction de $\mathbf{x}$.

Voisinages

Définition

Definition

Un voisinage centré en $\mathbf{x}$ est un sous-ensemble $\mathcal{V}_{\mathbf{x}} \subset \mathbf{E}$, déterminé par une fonction de $\mathbf{x}$.

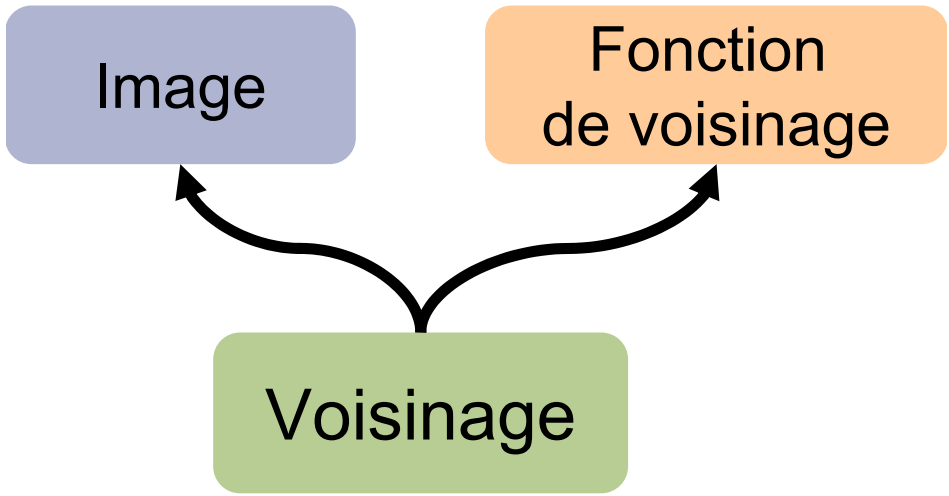
Les itérateurs sont un moyen générique pour décrire ce sous-ensemble

Voisinages

Définition

Definition

Un voisinage centré en **x** est un sous-ensemble $\mathcal{V}_{\mathbf{x}} \subset \mathbf{E}$, déterminé par une fonction de **x**.

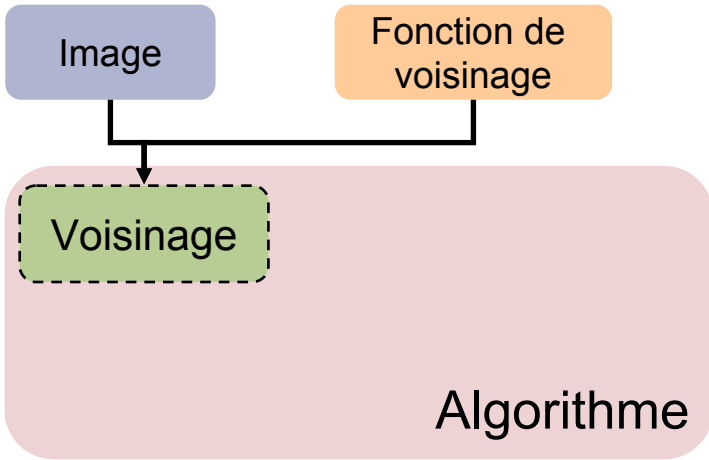


Voisinages

Fonctions de voisinage

Cas d'utilisation

1. Initialisation du voisinage (image, fonction de voisinage)

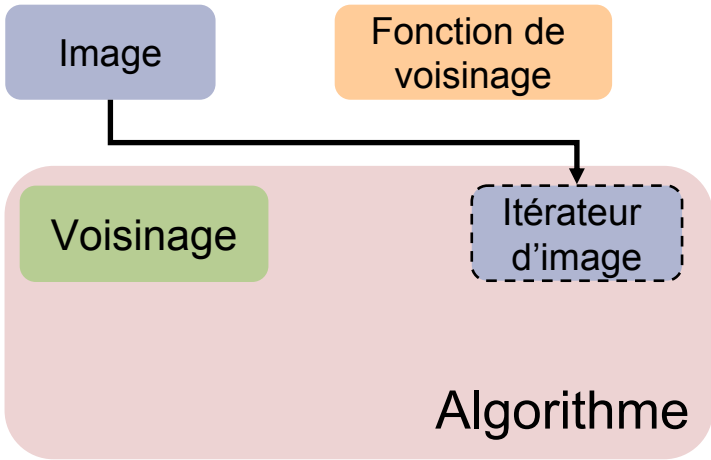


Voisinages

Fonctions de voisinage

Cas d'utilisation

1. Initialisation du voisinage (image, fonction de voisinage)

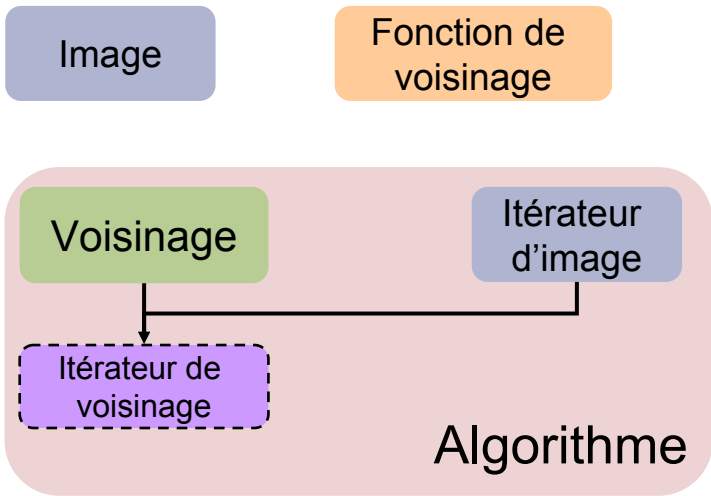


Voisinages

Fonctions de voisinage

Cas d'utilisation

1. Initialisation du voisinage (image, fonction de voisinage)
2. Centrage du voisinage

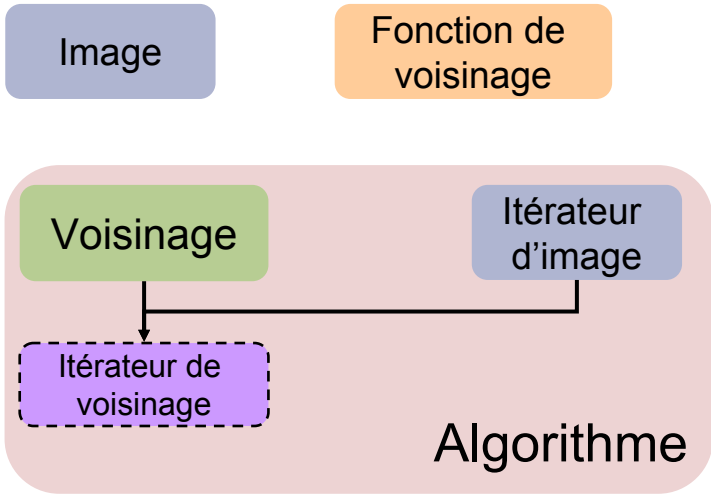


Voisinages

Fonctions de voisinage

Cas d'utilisation

1. Initialisation du voisinage (image, fonction de voisinage)
2. Centrage du voisinage
3. Récupération de l'ensemble des voisins
4. Retour en 2

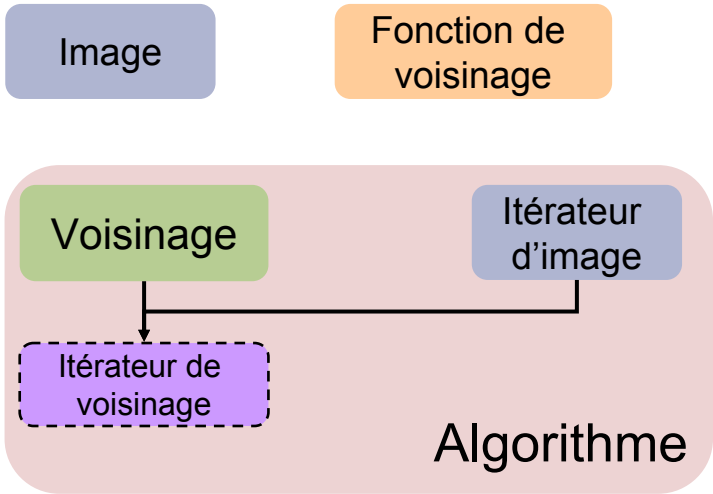


Voisinages

Fonctions de voisinage

Cas d'utilisation

1. Initialisation du voisinage (image, fonction de voisinage)
2. Centrage du voisinage
3. Récupération de l'ensemble des voisins
4. Retour en 2



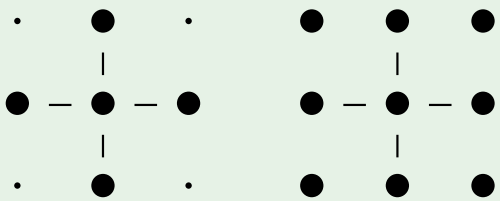
Conséquences de l'itérateur

1. Gestion de la topologie interne au voisinage
2. Algorithmes indépendants du type de voisinage utilisé

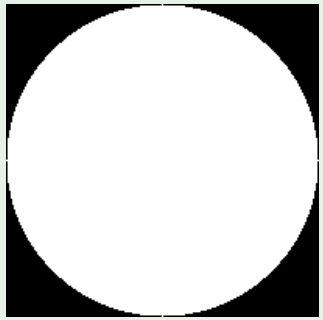
Voisinages

Fonctions de voisinage

Exemples de fonctions de voisinage



4 et 8 connexes $2D$



Fonction plus riche

Voisinages

Fonctions structurantes

Changement de la topologie de l'image par une fonction structurante

Éléments structurants suivant le mouvement

Nicolas Laveau & Christophe Bernard.
Structuring elements following the optical flow.
Proceedings of the 7th ISMM, 2005.

Voisinages

Fonctions structurantes

Changement de la topologie de l'image par une fonction structurante

Amibes morphologiques

Romain Lerallut, Étienne Decencière & Fernand Meyer.
Image filtering using Morphological Amoebas.
Proceedings of the 7th ISMM, 2005.

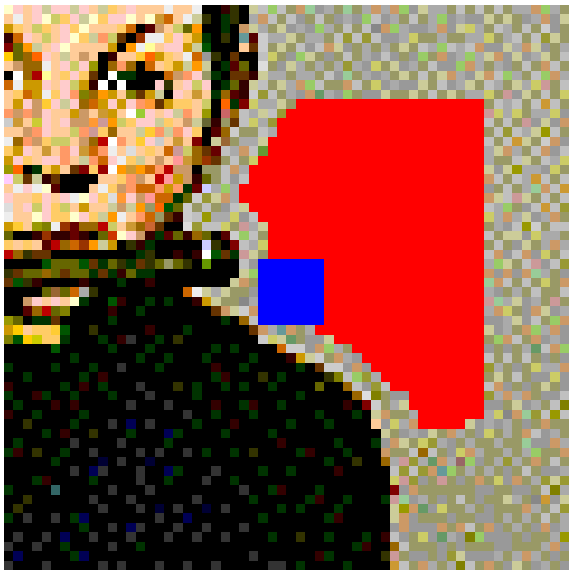
Voisinages

Fonctions structurantes

Changement de la topologie de l'image par une fonction structurante

Amibes morphologiques

Romain Lerallut, Étienne Decencière & Fernand Meyer.
Image filtering using Morphological Amoebas.
Proceedings of the 7th ISMM, 2005.



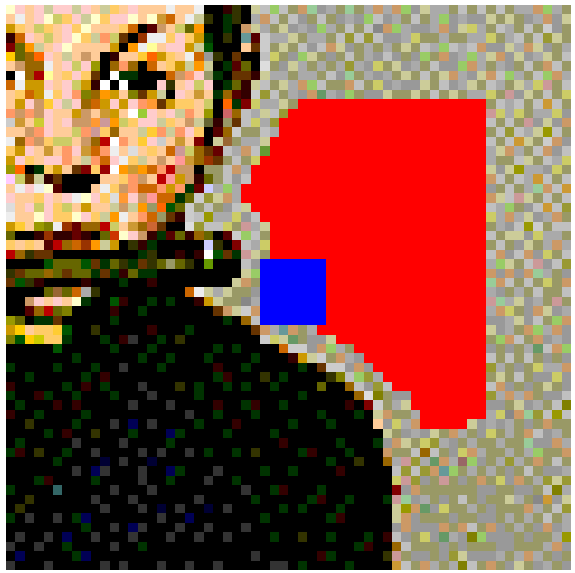
Voisinages

Fonctions structurantes

Changement de la topologie de l'image par une fonction structurante

Amibes morphologiques

Romain Lerallut, Étienne Decencière & Fernand Meyer.
Image filtering using Morphological Amoebas.
Proceedings of the 7th ISMM, 2005.



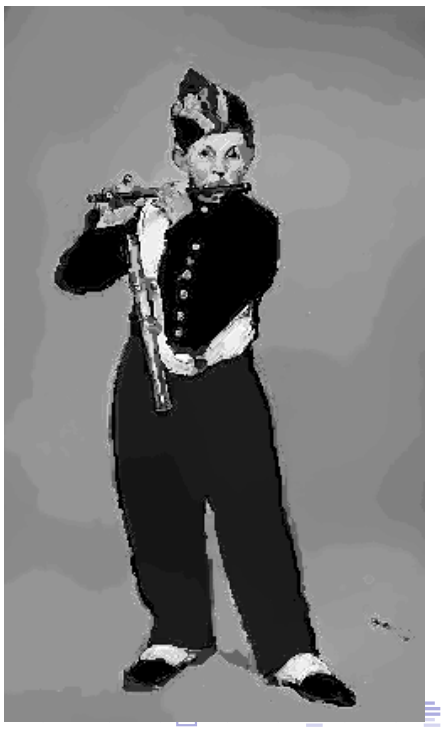
Voisinages

Fonctions structurantes

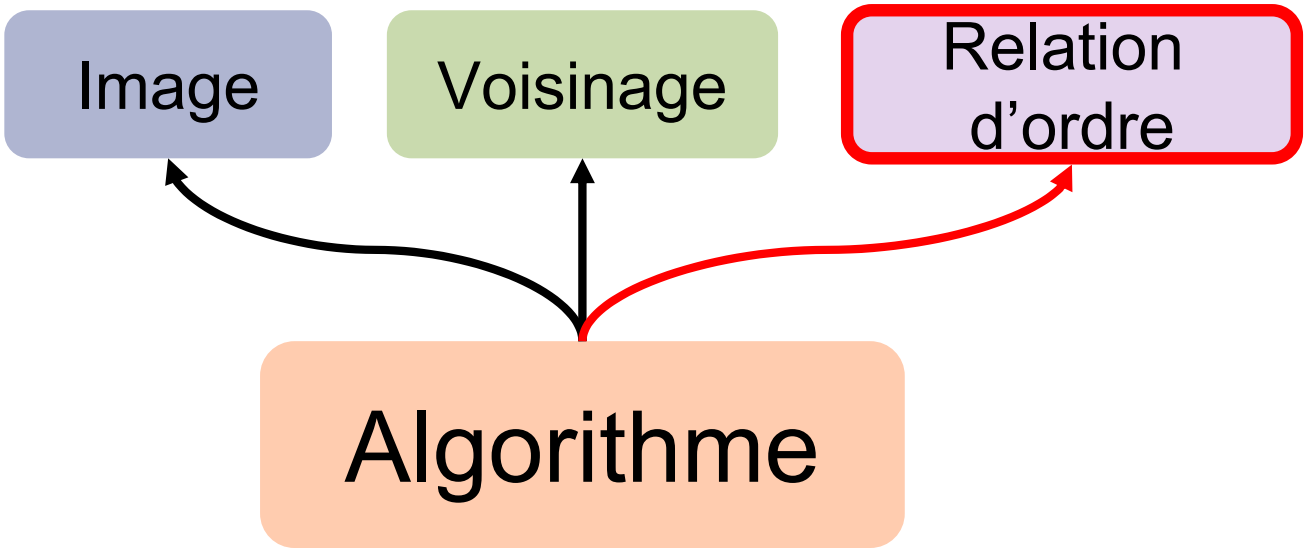
Changement de la topologie de l'image par une fonction structurante

Amibes morphologiques

Romain Lerallut, Étienne Decencière & Fernand Meyer.
Image filtering using Morphological Amoebas.
Proceedings of the 7th ISMM, 2005.



Algorithmes & Ordres



Treillis

Définition

Definition (Treillis)

Un treillis $\mathcal{L}_{\mathbf{F}} = \langle F, \preceq \rangle$ est un ensemble *ordonné* tel qu'il existe un plus petit majorant et un plus grand minorant, dans F, pour chaque paire d'éléments.

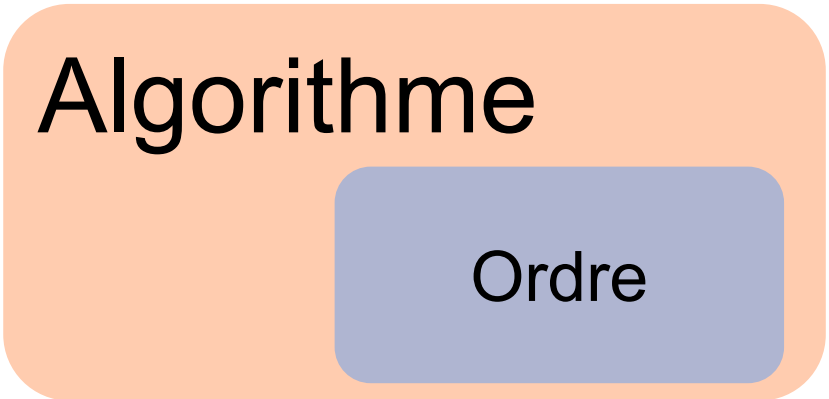
Treillis

Définition

Definition (Treillis)

Un treillis $\mathcal{L}_{\mathbf{F}} = \langle F, \preceq \rangle$ est un ensemble *ordonné* tel qu'il existe un plus petit majorant et un plus grand minorant, dans F, pour chaque paire d'éléments.

Nécessité d'avoir une relation d'ordre sur tous les espaces **F** possibles



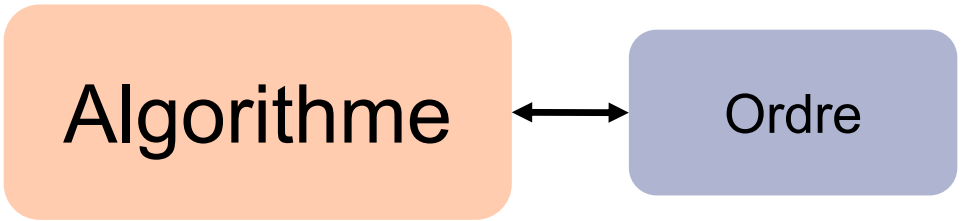
Treillis

Définition

Definition (Treillis)

Un treillis $\mathcal{L}_{\mathbf{F}} = \langle F, \preceq \rangle$ est un ensemble *ordonné* tel qu'il existe un plus petit majorant et un plus grand minorant, dans F, pour chaque paire d'éléments.

$\preceq$ est simplement un opérateur booléen à deux entrées !

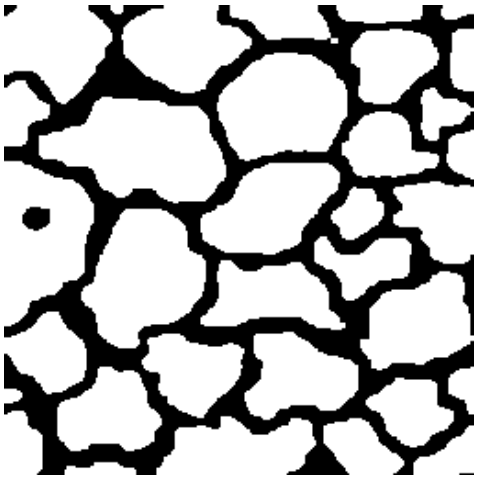


Aspects multidimensionnels & multispectraux

Multidimensionnalité

Dimension des images

- Images en 2 dimensions

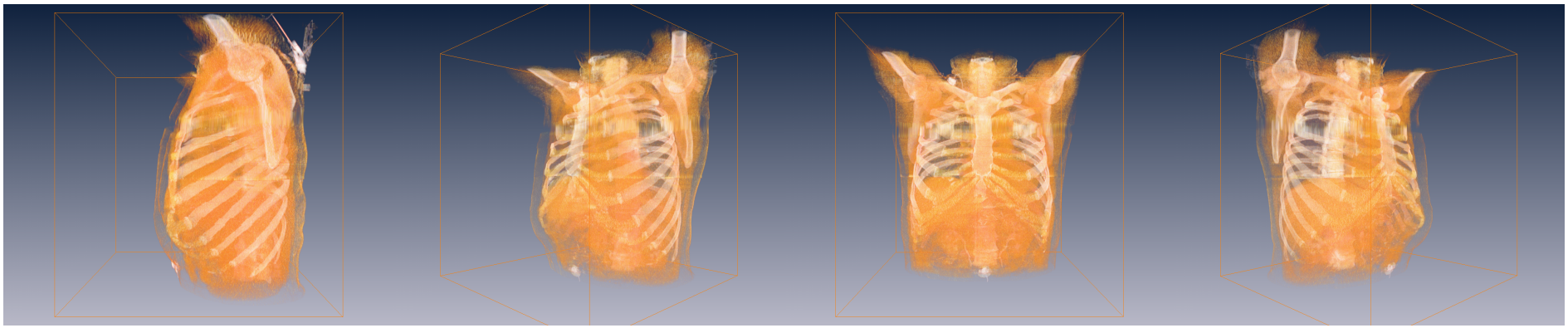


Aspects multidimensionnels & multispectraux

Multidimensionnalité

Dimension des images

- Images en 2 dimensions
- Images en 3 dimensions



Aspects multidimensionnels & multispectraux

Multidimensionnalité

Dimension des images

- Images en 2 dimensions
- Images en 3 dimensions

Images de dimension supérieure

- Ajout d'une dimension temporelle aux images $3D$
- Synthétisation/génération d'images

Aspects multidimensionnels & multispectraux

Multidimensionnalité

Dimension des images

- Images en 2 dimensions
- Images en 3 dimensions

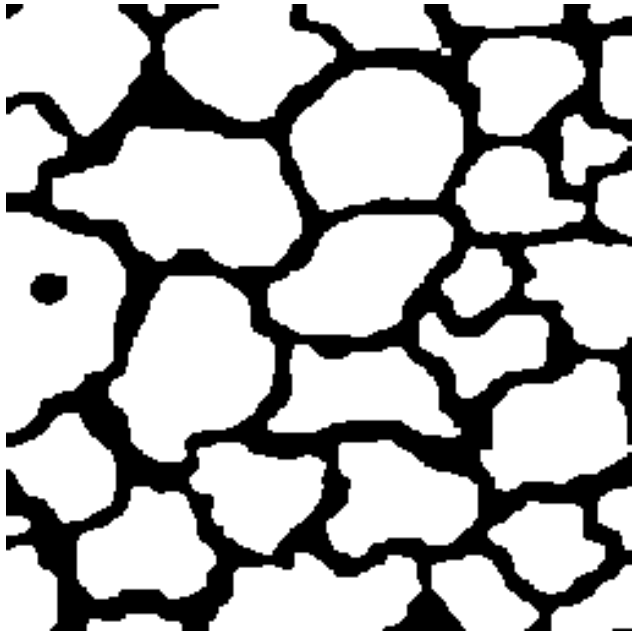
Aucune restriction (algorithmique) des structures à un type d'image !

Aspects multidimensionnels & multispectraux

Multispectralité

Types d'image

- Binaire



Aspects multidimensionnels & multispectraux

Multispectralité

Types d'image

- Binaire
- Scalaire

Aspects multidimensionnels & multispectraux

Multispectralité

Types d'image

- Binaire
- Scalaire
- Couleur

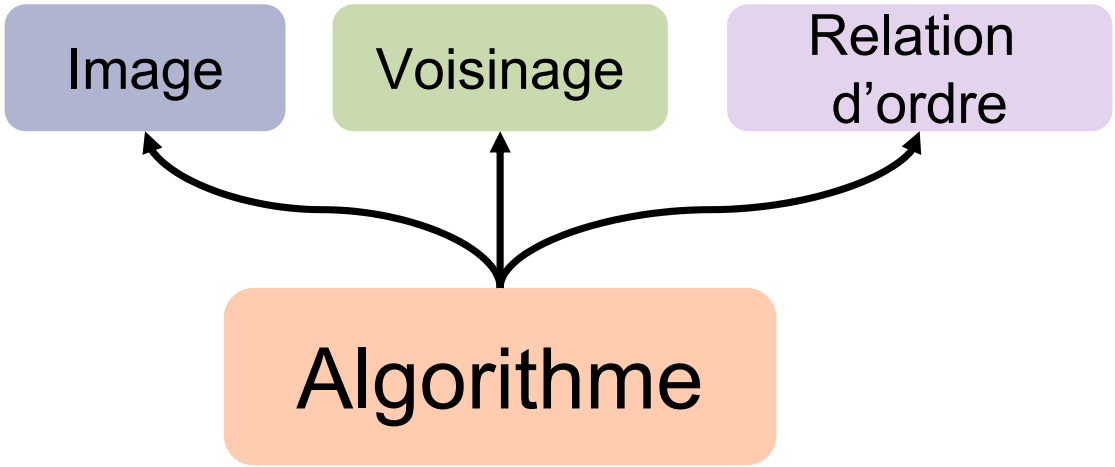
Aspects multidimensionnels & multispectraux

Multispectralité

Types d'image

- Binaire
- Scalaire
- Couleur
- Multispectral

Applications aux algorithmes



Utilisation

Exemple de l'érosion (élément structurant plan)

Érosion

La destination reçoit l'infimum du voisinage centré sur chaque site de la source.

Utilisation

Exemple de l'érosion (élément structurant plan)

Érosion

La destination reçoit l'infimum du voisinage centré sur chaque site de la source.

Inventaire

1. Image d'entrée $\mathcal{I}$

Utilisation

Exemple de l'érosion (élément structurant plan)

Érosion

La destination reçoit l'infimum du voisinage centré sur chaque site de la source.

Inventaire

1. Image d'entrée $\mathcal{I}$
2. $\forall p \in \mathbf{E}$

Utilisation

Exemple de l'érosion (élément structurant plan)

Érosion

La destination reçoit l'infimum du voisinage centré sur chaque site de la source.

Inventaire

1. Image d'entrée $\mathcal{I}$
2. $\forall p \in \mathbf{E}$
3. $\forall v \in \mathcal{N}_p$

Utilisation

Exemple de l'érosion (élément structurant plan)

Érosion

La destination reçoit l'infimum du voisinage centré sur chaque site de la source.

Inventaire

1. Image d'entrée $\mathcal{I}$
2. $\forall p \in \mathbf{E}$
3. $\forall v \in \mathcal{N}_p$
4. Image de sortie $\mathcal{J}$

Utilisation

Exemple de l'érosion (élément structurant plan)

Érosion

La destination reçoit l'infimum du voisinage centré sur chaque site de la source.

Inventaire

1. Image d'entrée $\mathcal{I}$
2. $\forall p \in \mathbf{E}$
3. $\forall v \in \mathcal{N}_p$
4. Image de sortie $\mathcal{J}$
5. Opérateur d'infimum

Utilisation

Exemple de l'érosion (élément structurant plan)

Érosion

La destination reçoit l'infimum du voisinage centré sur chaque site de la source.

Inventaire

1. Image d'entrée $\mathcal{I}$
2. $\forall p \in \mathbf{E}$
3. $\forall v \in \mathcal{N}_p$
4. Image de sortie $\mathcal{J}$
5. Opérateur d'infimum

Remarque

Opérateur $\bigwedge$ est un paramètre d'un méta-algorithme générique.

Utilisation

Érosions et dilatations lexicographiques

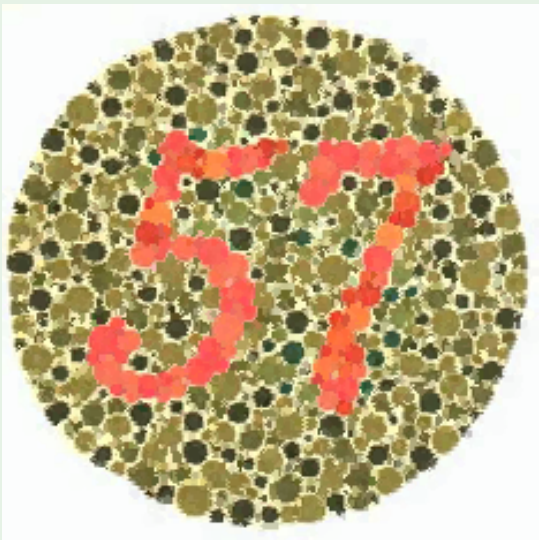
Ordre lexicographique $\preceq$

$\preceq$: ordre sur un espace vectoriel

Dilatations et érosions lexicographiques

Image originale

Dilaté $\preceq_{S\downarrow L\downarrow H\downarrow}$



Érodé $\preceq_{S\downarrow L\downarrow H\downarrow}$

Ligne de partage des eaux $4D$

Exemple $3D + t$

Gestion des images multidimensionnelles

LPE « $4D$ »

Volume $3D + t$
Marquage du poumon
Marqueur dans une coupe $2D$ du temps 0

Raffi Enficiaud.
Algorithmes multidimensionnels et multispectraux en morphologie mathématique : approche par méta-programmation..
Centre de Morphologie Mathématique, ENSMP, 2007.

Labélisation

Description

Objectif de l'algorithme

Identifier toutes les composantes connexes.

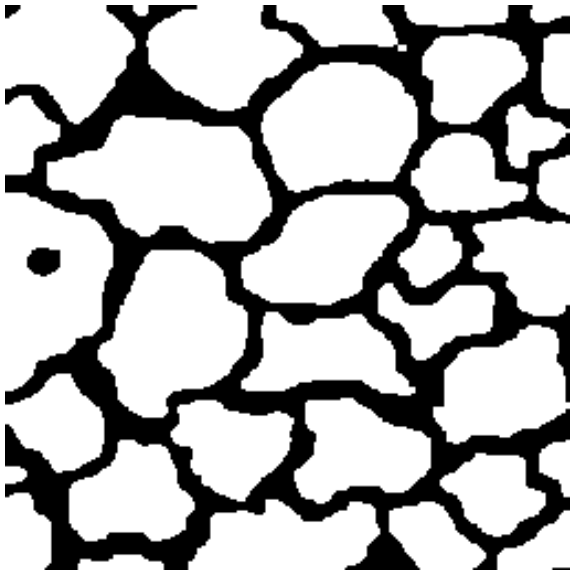


Image originale

Image labélisée

Labélisation

Description

Objectif de l'algorithme

Identifier toutes les composantes connexes.

Composante connexe ?

1. Notion de connexité

Labélisation

Description

Objectif de l'algorithme

Identifier toutes les composantes connexes.

Composante connexe ?

1. Notion de connexité
2. Même composante ?
   - Fond/forme

Labélisation

Description

Objectif de l'algorithme

Identifier toutes les composantes connexes.

Composante connexe ?

1. Notion de connexité
2. Même composante ?
   - Fond/forme
   - Zone plate

Labélisation

Description

Objectif de l'algorithme

Identifier toutes les composantes connexes.

Composante connexe ?

1. Notion de connexité
2. Même composante ?
   - Fond/forme
   - Zone plate
   - Zone λ-plate

Labélisation

Description

Objectif de l'algorithme

Identifier toutes les composantes connexes.

Composante connexe ?

1. Notion de connexité
2. Même composante ?
   - Fond/forme
   - Zone plate
   - Zone λ-plate
   - ...

Labélisation

Formulation théorique

Labélisation : formulation

Ensemble quotient d'une relation d'équivalence $=_{\mathcal{R}}$

Labélisation

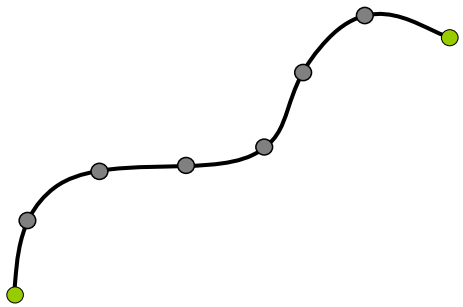
Formulation théorique

Labélisation : formulation

Ensemble quotient d'une relation d'équivalence $=_{\mathcal{R}}$

1. Connexion

   $\mathbf{x}_a$ et $\mathbf{x}_b$ sont connectés par un arc



Labélisation

Formulation théorique

Labélisation : formulation

Ensemble quotient d'une relation d'équivalence $=_{\mathcal{R}}$

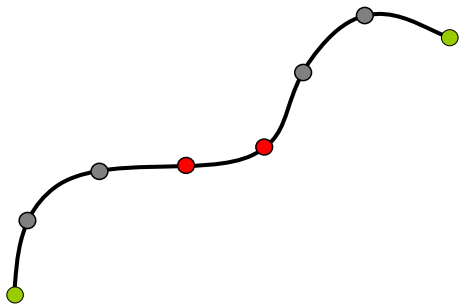
1. Connexion

   $\mathbf{x}_a$ et $\mathbf{x}_b$ sont connectés par un arc

2. Valeur de l'image des points

   tous les points voisins de cet arc vérifient une relation $\mathcal{R}_I$

$$\mathbf{x}_1 \mathcal{R}_I \mathbf{x}_2 \Leftrightarrow \begin{cases} \\ \ldots \end{cases}$$



Labélisation

Formulation théorique

Labélisation : formulation

Ensemble quotient d'une relation d'équivalence $=_{\mathcal{R}}$

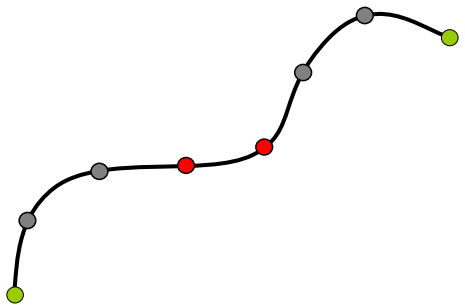
1. Connexion

   $\mathbf{x}_a$ et $\mathbf{x}_b$ sont connectés par un arc

2. Valeur de l'image des points

   tous les points voisins de cet arc vérifient une relation $\mathcal{R}_I$

$$\mathbf{x}_1 \mathcal{R}_I \mathbf{x}_2 \Leftrightarrow \begin{cases} \mathcal{I}(\mathbf{x}_1) = \mathcal{I}(\mathbf{x}_2) \\ \\ \dots \end{cases}$$



Labélisation

Formulation théorique

Labélisation : formulation

Ensemble quotient d'une relation d'équivalence $=_{\mathcal{R}}$

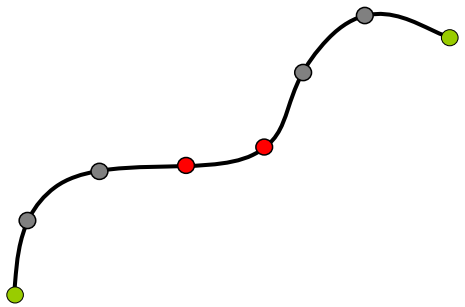
1. Connexion

   $\mathbf{x}_a$ et $\mathbf{x}_b$ sont connectés par un arc

2. Valeur de l'image des points

   tous les points voisins de cet arc vérifient une relation $\mathcal{R}_I$

$$\mathbf{x}_1 \mathcal{R}_I \mathbf{x}_2 \Leftrightarrow \begin{cases} \mathcal{I}(\mathbf{x}_1) = \mathcal{I}(\mathbf{x}_2) \\ \|\mathcal{I}(\mathbf{x}_1) - \mathcal{I}(\mathbf{x}_2)\|_{\mathbf{F}} \leq \lambda \\ \dots \end{cases}$$



Labélisation

Formulation théorique

Labélisation : formulation

Ensemble quotient d'une relation d'équivalence $=_{\mathcal{R}}$

1. Connexion

   $\mathbf{x}_a$ et $\mathbf{x}_b$ sont connectés par un arc

2. Valeur de l'image des points

   tous les points voisins de cet arc vérifient une relation $\mathcal{R}_I$

$$\mathbf{x}_1 \mathcal{R}_I \mathbf{x}_2 \Leftrightarrow \begin{cases} \mathcal{I}(\mathbf{x}_1) = \mathcal{I}(\mathbf{x}_2) \\ \|\mathcal{I}(\mathbf{x}_1) - \mathcal{I}(\mathbf{x}_2)\|_{\mathbf{F}} \leq \lambda \\ \dots \end{cases}$$

Formulation adaptée à la méta-programmation !

Labélisation

Formulation théorique

Labélisation : formulation

Ensemble quotient d'une relation d'équivalence $=_{\mathcal{R}}$

1. Connexion

   $\mathbf{x}_a$ et $\mathbf{x}_b$ sont connectés par un arc

2. Valeur de l'image des points

   tous les points voisins de cet arc vérifient une relation $\mathcal{R}_I$

$$\mathbf{x}_1 \mathcal{R}_I \mathbf{x}_2 \Leftrightarrow \begin{cases} \mathcal{I}(\mathbf{x}_1) = \mathcal{I}(\mathbf{x}_2) \\ \|\mathcal{I}(\mathbf{x}_1) - \mathcal{I}(\mathbf{x}_2)\|_{\mathbf{F}} \leq \lambda \\ \dots \end{cases}$$

Quelques lignes suffisent pour définir une nouvelle labélisation à l'aide des outils de méta-programmation !

Généricité & Spécialisation

Universalité des structures ralentissant globalement l'exécution

Généricité & Spécialisation

Universalité des structures ralentissant globalement l'exécution

Solution ?

Mécanisme de spécialisation

Généricité & Spécialisation

Universalité des structures ralentissant globalement l'exécution

Solution ?

Mécanisme de spécialisation

Optimisations

Jaromír Brambor.
Algorithmes de la Morphologie Mathématique pour les architectures orientées flux.
Centre de Morphologie Mathématique, ENSMP, 2006.

« Morph-M » (ie. Morphée)

Librairie de traitement d'image et de morphologie mathématique utilisant la généricité

- En constante évolution
- Permettant un travail collaboratif
- Incluant des implémentations algorithmiques de référence
- Puissante
  - Algorithmes « indépendants » de la représentation des données
  - Optimisations pour des cas particuliers d'image et d'architecture matérielle
- Modulaire
- Utilisée dans de nombreux projets (industriels ou de recherche)

Synthèse

Avantages

- Cadre étendant le champ d'application des algorithmes
- Récupération simple des algorithmes existants sans modification importante
- $\frac{\text{performances}}{\text{coûts de développement}} >> 1$

Synthèse

Avantages

- Cadre étendant le champ d'application des algorithmes
- Récupération simple des algorithmes existants sans modification importante
- $\frac{\text{performances}}{\text{coûts de développement}} >> 1$

Inconvénients

- mais ...

Morph-M et généricité en traitement d'images.

Part 2 : les inconvénients du générique

Inconvénients

There is no silver bullet

Constat :

1. Les approches génériques sont fabuleuses
2. Rien de plus dangereux qu'un développeur fou avec un jouet fabuleux ;)

Où, quand, comment, pourquoi, combien ?

Les questions que votre manager/chef/TypeQuiVousFinance vous posera un jour...

A savoir aussi :

Quand ne PAS utiliser d'approche générique...

Plan

Les coûts cachés

Représentativité ?

- Quels problème *peut-on* résoudre vs
- Quels problème *doit-on* résoudre !

Dilemme

- Peut-on faire du générique sans moult exemples de problèmes ?
- Combien d'exemples suffisent pour généraliser ?

(réponses : 1. ça dépend, 2. un certain nombre)

Les coûts cachés

L'usine à gaz

- temps de développement
  - coût !
  - "downtime" vs incrémentalité des fonctionnalités
- performance (vitesse)
  - doit faire partie des specs *initiales*
  - à contrôler en permanence
- taille du code
  - navigation
  - compilation

Les coûts cachés

Les gens : utilisateurs, clients, autres développeurs

- Concepts plus abstraits donc moins simples à saisir
- Temps d'adaptation ("stagiaire-friendly"?)
- Coût d'un problème simple

Programmation conceptuelle puissante, mais...

Différences importantes en termes de « performances »

- le compilateur gère virtuellement toutes les combinaisons (lourd à la compilation)

Programmation conceptuelle puissante, mais...

Différences importantes en termes de « performances »

- le compilateur gère virtuellement toutes les combinaisons (lourd à la compilation)
- approche par union : les nouvelles spécialisations s'ajoutent aux anciennes ... sauf si le projet a été mal conçu (arrive souvent) ou si besoin de nouvelles fonctionnalités (arrive très souvent).

Programmation conceptuelle puissante, mais...

Différences importantes en termes de « performances »

- le compilateur gère virtuellement toutes les combinaisons (lourd à la compilation)
- approche par union : les nouvelles spécialisations s'ajoutent aux anciennes ... sauf si le projet a été mal conçu (arrive souvent) ou si besoin de nouvelles fonctionnalités (arrive très souvent).
- Implémentation peut-être optimale... À une constante près

Performances des outils

Le compilateur

Différences importantes en termes de « performances »

- Visual 6 : sans commentaires
- Visual 7 : ok, mais code généré de pas très bonne qualité
- Visual 8 : plus « standard » que Visual 7, code meilleurs mais à améliorer
- GCC 3.4 : très proche du standard
- GCC 4 : encore plus proche du standard, mais code généré assez lent
- GCC : développements lents
- Intel IPP 7 : bonne qualité de code généré, mais non standard
- Intel IPP 8 : standard mais très très lent.
- Intel IPP 9 : pas testé

Performances des outils

Outils de conception

Quels outils s'offrent au programmeur ?

- Outils de traitement de texte

Performances des outils

Outils de conception

Quels outils s'offrent au programmeur ?

- Outils de traitement de texte
- Papier et crayon...

Performances des outils

Outils de conception

Quels outils s'offrent au programmeur ?

- Outils de traitement de texte
- Papier et crayon...
- Aucun outil de conceptions ?

Synthèse

Un outil à maitriser

Tenir compte des risques

Le code est meilleur et plus compact mais :

- risque d'*hubris* (donc perte de focalisation)
- problèmes de lourdeur (design, dev, exécution)

Risque majeur :

Quand on est lancés, pourquoi s'arrêter à ce dont on a besoin ? !

Savoir-faire

Pour être performant, il faut :

- maitriser le plus grand nombre de “trucs”
- en connaitre les avantages/inconvénients
- spécifier dès le début (notamment les perfs)

Et pour ça

le mieux est d’en faire beaucoup et d’en jeter beaucoup. Itérer.

La généricité “light”

“design for change”

- commencer simple (YAGNI)
- rendre le code generic-friendly
- designer pour l’évolutif

L’important

c’est de se fermer le moins d’options, tout en restant optimal sur ce qu’on fait *maintenant*.

Merci de votre attention

Release early, release often...