

Master IMA - UMPC Paris 6
RDMM - Année 2009-2010
Fiche de TP

1 Préliminaires

1. Récupérez l'archive du logiciel de TP à partir du lien suivant :
`http://www.ensta.fr/~manzaner/Cours/IMA/tp2009.tar`
2. Développez l'archive :
`>tar xvf tp2009.tar`
3. Compilez le logiciel :
`>cd TPMorpho/`
`>make`
4. Récupérez quelques images aux formats *gif* ou *pgm*, à partir du lien suivant :
`http://www.ensta.fr/~manzaner/Images`
5. Lancez le logiciel (l'argument donné ici n'est qu'un exemple) :
`>./Morpho.tcl -i ../IMAGES/clock.gif -nb 6 &`

2 Opérateurs de base et composés

EXPÉRIMENTEZ : A partir des opérations disponibles sur l'interface *Inti*, calculez les opérations morphologiques suivantes :

- Erosion
- Dilatation
- Gradient interne
- Gradient externe
- Gradient morphologique (symétrisé)
- Laplacien morphologique
- Ouverture
- Fermeture
- Top-hat
- Top-hat conjugué

Quel est l'effet du changement de la condition de bord dans l'interface sur les opérations morphologiques ?

Comment vérifier expérimentalement la relation de dualité des opérateurs (e.g. Erosion/Dilatation, Ouverture/Fermeture) ?

3 Un détecteur de contours rudimentaire

Selon le modèle classique de Marr et Hildreth, les *contours* d'une image correspondent aux lieux de passage par zéro du *laplacien* Λ . On calcule cet ensemble à partir des 2 ensembles suivants :

$$\Lambda^+ = \{(x, y); \Lambda(x, y) > 0\}, \text{ et}$$

$$\Lambda^- = \{(x, y); \Lambda(x, y) < 0\}.$$

Ces 2 ensembles sont évidemment disjoints, mais on peut calculer grossièrement les lieux de passages par zéro par exemple de la façon suivante :

$$\Lambda^0 = \delta_B(\Lambda^+) \cap \Lambda^-,$$

où δ_B désigne la dilatation morphologique par une boule élémentaire (i.e. définissant la topologie).

Comme en général, ce détecteur fournit une réponse trop dense pour être exploitable, on le combine avec un seuillage *du module du gradient* G , soit :

$$G_s = \{(x, y); G(x, y) \geq s\}.$$

L'ensemble des *contours* C est finalement défini par :

$$C = G_s \cap \Lambda^0$$

APPLIQUEZ ce détecteur de contours directement sur l'interface *Inti* en prenant pour Λ et pour G respectivement le laplacien et le gradient morphologiques, et en testant sur 2 ou 3 images en niveaux de gris.

Quels sont les paramètres de votre détecteur ? Quel est l'impact de ces paramètres sur le résultat ?

Quels sont selon vous les limites à la pertinence et à l'exploitation de ce détecteur ?

4 Transformée en Tout-ou-rien.

Etant donné une image binaire $I \subset \mathbb{Z}^2$, un couple (H, M) d'éléments structurants $H, M \subset \mathbb{Z}^2$ tels que $H \cap M = \emptyset$, on appelle *transformée en tout-ou-rien* (TTR) de I par le couple (H, M) , l'image :

$$I \otimes (H, M) = \varepsilon_H(I) \cap \varepsilon_M(I^c) = (I \ominus \check{H}) \cap (I^c \ominus \check{M}).$$

EXPÉRIMENTEZ la recherche de configurations sur images binaires par TTR sur l'interface *Inti* (Sur le tableau de définition de l'élément structurant, le bouton de gauche définit H (en blanc), le bouton de droite définit M (en noir)).

Appliquez les TTR au débruitage de l'image binaire *texte_bruit.gif*, commentez les résultats.

5 Opérateurs connexes

EXPÉRIMENTEZ la reconstruction géodésique, pour une image binaire, puis pour une image en niveau de gris. Comment peut-on "calculer" une reconstruction sur l'interface *Inti* sans utiliser le bouton "reconstruction" ? Comment cet algorithme est-il programmé efficacement ?

APPLIQUEZ la reconstruction géodésique à l'image binaire *particules.gif* pour éliminer les particules de "diamètre" inférieur à 20 pixels sans modifier les autres. Quelle est l'opération que vous avez utilisée ?

CALCULEZ l'ouverture et la fermeture par reconstruction de l'image *tree_noise.gif*. Présentez et commentez les résultats.

EXPÉRIMENTEZ l'opération d'érosion ultime sur une image binaire. Quel rapport existe-il entre cette opération et la reconstruction géodésique? Appliquez l'érosion ultime à la singularisation de particules dans l'image binaire *coffee.gif*.

6 Lignes de Partage des Eaux

Expérimentez l'opérateur de lignes de partage des eaux (LPE - Menu "niveau de gris"). Comment interpréter cet opérateur? Sur quels types d'images l'applique-t-on en général?

Observez et interpréter le phénomène de sur-segmentation sur une image naturelle.

Appliquez une fermeture par reconstruction sur l'image de gradient avant la LPE. Quel est l'effet de cet opérateur sur le résultat de la LPE?

Appliquez un filtrage de dynamique sur l'image originale avant le calcul du gradient. Appliquez le gradient, puis la LPE. Quel est l'effet de l'opérateur de filtrage de dynamique?

Combinez enfin les 2 opérateurs avant le calcul de la LPE et interprétez le résultat.

Attention : la ligne de partage des eaux est affichée en jaune sur l'interface, mais sera interprétée en niveau de gris si on applique un opérateur *après la LPE*, ce qui n'aura donc pas de sens en général.

7 Programmation : Erosion binaire de taille arbitraire

7.1 Première version

PROGRAMMEZ la procédure :

```
void Erosion_version1(Image<int>& p,int distance,int rayon);
```

qui calcule l'érosion d'une image p codée sur des niveaux de gris allant de 0 à 255, pour un élément structurant égal à la boule discrète $d_{distance}$ ($distance = 4$ ou 8), de rayon $rayon$, en itérant $rayon$ fois une érosion élémentaire par une boule élémentaire de la même distance (voir Figure 1).

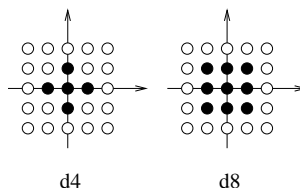


FIG. 1 – Les boules élémentaires des distances d_4 et d_8 .

Quelle est la complexité de votre algorithme en fonction de $rayon$?

7.2 Seconde version

EXPÉRIMENTEZ les transformées en distance sur une image binaire. Comment une telle transformation peut-elle être utilisée pour calculer des érosions binaires de taille arbitraire ?

La procédure

```
void FctDistance(Image<int>& p, char *nom);
```

disponible dans les sources du programme *Inti*, dans le fichier *morfo.cc*, calcule la transformée en distance *nom* de l'image binaire *p*, c'est-à-dire la fonction qui à chaque pixel non nul de l'image binaire (pixel appartenant à l'image au sens ensembliste), associe sa distance (au sens de *nom*) au plus proche pixel nul de l'image binaire (pixel appartenant au complémentaire de l'image au sens ensembliste). Cette transformée se calcule par deux balayages successifs sur l'image, l'un dans le sens video, l'autre dans le sens video inverse, et par application des masques de calcul des différentes distances discrètes (voir Figure 2). Pour chaque parcours, à chaque pixel non nul examiné, on remplace la valeur du pixel par le minimum de la valeur de ses "voisins" au sens du masque de parcours, augmenté de la valeur correspondante dans le masque. Après les deux parcours, la valeur des pixels est rééchelonnée sur l'intervalle $[0, 255]$ pour des raisons de visualisation.

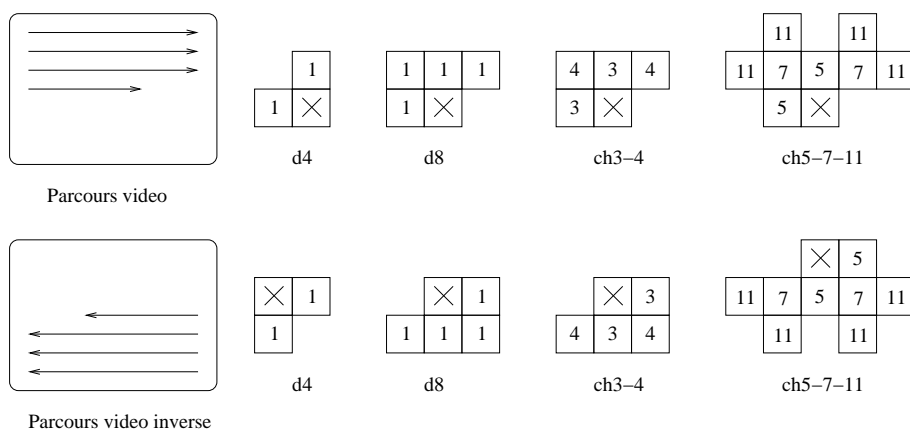


FIG. 2 – Masques de calcul utilisés pour les transformées en distance d_4 et d_8 , et pour les distances de chanfrein 3-4 et 5-7-11.

En fait, une procédure récursive similaire peut être utilisée pour calculer une bonne approximation de la transformée en distance *euclidienne* associée à une image binaire. Le principe, dû à Danielsson et Leymarie, consiste à calculer en chaque pixel le carré de la distance euclidienne exacte, en calculant récursivement pour chaque pixel (x, y) les *coordonnées relatives* $(R_x(x, y), R_y(x, y))$ du pixel de contour le plus proche (c'est-à-dire que le pixel du contour le plus proche de (x, y) est le pixel $(x + R_x(x, y), y + R_y(x, y))$). Le carré de la transformée en distance euclidienne est alors donné par $R_x(x, y)^2 + R_y(x, y)^2$. La fonction distance au carré est calculée récursivement par sommation marginale (i.e. quand un nombre r augmente de 1, son carré augmente de $2r + 1$). L'algorithme précis est le suivant :

Initialisation

pour tout pixel (x, y) :

si $(x, y) \notin X : \{f(x, y) = 0; R_x(x, y) = 0; R_y(x, y) = 0;\}$

si $(x, y) \in X : \{f(x, y) = \infty; R_x(x, y) = 0; R_y(x, y) = 0;\}$

Balayage direct

pour y de 0 à N_{lig} :

pour x de 0 à N_{col} :

(1) $(a, b) = \arg \min_{(a,b) \in V^-} [f(x+a, y+b) + \Delta f^{(a,b)}(x, y)]$

avec $\Delta f^{(a,b)}(x, y) = 2(|aR_x(x+a, y+b)| + |bR_y(x+a, y+b)|) + a^2 + b^2$

(2) $R_x(x, y) = R_x(x+a, y+b) + a$

$R_y(x, y) = R_y(x+a, y+b) + b$

(3) $f(x, y) = f(x+a, y+b) + \Delta f^{(a,b)}(x, y)$

Balayage rétrograde

pour y de N_{lig} à 0 :

pour x de N_{col} à 0 :

(1) $(a, b) = \arg \min_{(a,b) \in V^+} [f(x+a, y+b) + \Delta f^{(a,b)}(x, y)]$

avec $\Delta f^{(a,b)}(x, y) = 2(|aR_x(x+a, y+b)| + |bR_y(x+a, y+b)|) + a^2 + b^2$

(2) $R_x(x, y) = R_x(x+a, y+b) + a$

$R_y(x, y) = R_y(x+a, y+b) + b$

(3) $f(x, y) = f(x+a, y+b) + \Delta f^{(a,b)}(x, y)$

$f(x, y)$ correspond au carré de la transformée en distance euclidienne.

$R_x(x, y)$ et $R_y(x, y)$ correspondent respectivement aux coordonnées relatives horizontale et verticale du pixel de contour le plus proche de (x, y) .

$V^- = \{(-1, -1), (0, -1), (+1, -1), (-1, 0)\}$, et

$V^+ = \{(+1, 0), (-1, +1), (0, +1), (+1, +1)\}$

correspondent respectivement aux voisinages causal et anticausal.

La procédure

```
void Fct_distance_euclidienne(Image<int>& p);
```

disponible dans le fichier *morfo.cc*, calcule de cette façon la transformée en distance euclidienne de l'image binaire *p*.

ADAPTEZ les procédures précédentes pour programmer la procédure :

```
void Erosion_version2(Image<int>& p, char *distance, int rayon);
```

qui calcule l'érosion d'une image binaire *p*, pour une boule de rayon *rayon* de la distance *distance*.

Quelle est la complexité de votre algorithme en fonction de *rayon* ?

8 Programmation : Nivellement

Le Nivellement peut être vu comme une généralisation de la reconstruction, utilisée dans le cas où il n'y a pas de relation d'ordre entre le marqueur et l'image de référence.

PROGRAMMEZ en utilisant la procédure de reconstruction géodésique disponible dans le fichier *morfo.cc* de l'interface *Inti*, la procédure :

```
void Nivellement(Image<int>& p, Image<int>& p_ref, int cx);
```

qui calcule le nivellement de l'image marqueur p dans l'image de référence p_{ref} , en utilisant la connexité cx ($cx = 4$ ou 8).

EXPÉRIMENTEZ les nivellements pour calculer les filtres alternés séquentiels par reconstruction d'une image en niveaux de gris sur plusieurs niveaux. Observez l'évolution de profils 1d (bouton de droite sur les images) pour les f.a.s et les f.a.s par reconstruction. Commentez les résultats.

Les filtres alternés séquentiels peuvent être calculés "à la main" depuis l'interface *Inti*, ou, mieux, en programmant votre propre procédure :

```
void FAS(Image<int>& p,int premier,int rayon);
```

qui calcule le filtrage alterné séquentiel de l'image p par le filtre :

(ouverture d'abord) si $premier = 0$, et

(fermeture d'abord) si $premier = 1$.