TP de Traitement d'Images Médicales

Décembre 2006

Gwenaël Brunet & Laurent Lecornu
Département Image et Traitement de l'Information
Ecole Nationale Supérieure des Télécommunications de Bretagne (ENSTB)
Groupe des Ecoles des Télécommunications (GET)
Gwenael.Brunet@enst-bretagne.fr, Laurent.Lecornu@enst-bretagne.fr

Introduction

Ce TP vous propose de directement mettre en œuvre quelques concepts clefs de Traitement d'Images Numériques appliqués à diverses modalités d'Imagerie Médicale. Il est en prolongement direct du TP Traitement d'Images Numériques réalisé au cours de la deuxième année.

L'objectif principal de ce cours est de vous faire appliquer quelques traitements de base sur des images sélectionnées, afin de mieux comprendre comment ils fonctionnent, à quoi ils servent, et également de les mettre en application en implémentant ces algorithmes dans un langage couramment utilisé dans le milieu.

Rappels des principaux traitements

Opérateurs linéaires

Pour rappel, ces opérateurs sont implémentés en effectuant une convolution d'un masque sur une l'image traitée. Il existe différents masques, dont voici une partie :

Passe-bas (flou)

Filtre moyenneur	Filtre gaussien
$1/9\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}$	$1/16\begin{pmatrix}1&2&1\\2&4&2\\1&2&1\end{pmatrix}$

Passe-haut (contours)

	X	Y
Opérateur de Robert	$\begin{pmatrix}1&0\\0&-1\end{pmatrix}$	$\begin{pmatrix}0&1\\-1&0\end{pmatrix}$
Opérateur de Prewitt	$\begin{pmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{pmatrix}$	$\begin{pmatrix}-1&-1&-1\\0&0&0\\1&1&1\end{pmatrix}$
Opérateur de Sobel	$\begin{pmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{pmatrix}$	$\begin{pmatrix}-1&-2&-1\\0&0&0\\1&2&1\end{pmatrix}$

Filtre Laplacien		
$\begin{pmatrix}0&1&0\\1&-4&1\\0&1&0\end{pmatrix}$	$\begin{pmatrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{pmatrix}$	$\begin{pmatrix}-1&0&0&-1\\0&1&1&0\\0&1&1&0\\-1&0&0&-1\end{pmatrix}$

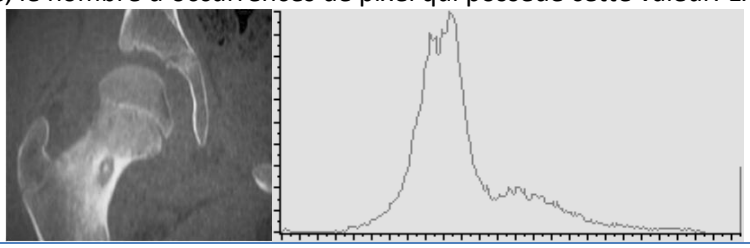
Opérateurs non linéaires

Filtrage médian	Le pixel étudié prend la valeur médiane de son voisinage. Par exemple : $\begin{bmatrix}150&55&82\\32&\mathbf{100}&65\\16&122&76\end{bmatrix}$ Tri: 16, 32, 55, 65, 76, 82, 100, 122, 150. Valeur médiane: 76. La valeur centrale 100 est donc remplacée par 76.
Érosion	Le pixel étudié prend la valeur minimale de son voisinage. Par exemple :

Dilatation	$\begin{bmatrix} 150 & 55 & 82 \\ 32 & \mathbf{100} & 65 \\ 16 & 122 & 76 \end{bmatrix}$	Tri: 16, 32, 55, 65, 76, 82, 100, 122, 150. Valeur minimale : 16. La valeur centrale 100 est donc remplacée par 16.
	Le pixel étudié prend la valeur maximale de son voisinage. Par exemple :	
	$\begin{bmatrix} 150 & 55 & 82 \\ 32 & \mathbf{100} & 65 \\ 16 & 122 & 76 \end{bmatrix}$	Tri: 16, 32, 55, 65, 76, 82, 100, 122, 150. Valeur maximale : 150. La valeur centrale 100 est donc remplacée par 150.

***Note** : il est coutume d'utiliser un enchaînement des opérateurs d'érosion et de dilatation. Lorsque l'on effectue une érosion suivie d'une dilatation, on parle d' « **Ouverture** », et à l'inverse, une dilatation suivie d'une érosion, on parle de « **Fermeture** ».*

Autres traitements

Contraste	Le contraste s'effectue par application d'un coefficient multiplicateur au pixel étudié. Lorsque ce coefficient est supérieur à 1, le contraste est augmenté, lorsqu'il est compris entre 0 et 1, le contraste est diminué.
Seuillage	Le seuillage consiste à mettre à zéro (0) tous les pixels de niveau de gris inférieur à un seuil (127 par exemple), et à 255 ceux qui sont supérieurs. On obtient ainsi une image en noir et blanc, soit deux niveaux.
Histogramme	L'histogramme permet d'obtenir un graphique qui décompte pour chaque valeur de niveau de gris, le nombre d'occurrences de pixel qui possède cette valeur. Exemple : 

Rappels de l'application en langage C

1. Caractéristiques de base des images numériques

Une image numérique est une matrice à deux dimensions, représentée du point de vue du langage C, par un tableau à deux dimensions "`[][]`" (double crochets).

Le présent TP ne traitera pas de la gestion optimisée à proprement parler des images numériques, du point de vue de la mémoire. Nous conviendrons donc, que chaque image correspond à un emplacement mémoire de taille prédéterminée (allocation de mémoire statique, par opposition à l'allocation de mémoire dynamique).

Définir une variable pour une image s'effectue par conséquent de la façon suivante :

```
unsigned char image[320][200];
```

Cette première déclaration permet de construire une image d'une taille de **320 pixels par 200**. Le type utilisé, « `unsigned char` », indique qu'il s'agit d'une image dont chaque pixel est codé sur 1 octet, soit 8 bits. 8 bits permettent de coder 2^8 niveaux, soit **256** (valeurs comprises entre 0 et 255, 0 représentant le noir, et 255 le blanc). L'image possède pour conclure, 320*200 pixels, codés chacun sur 256 niveaux (du gris dans notre cas).

Pour information, chaque image "pèsera" ainsi 64000 octets, soit 62.5 Ko.

Accéder à un pixel est relativement simple. Suivant la déclaration précédente, si l'on souhaite obtenir la valeur du pixel présent à la position (115, 28), il suffit d'écrire :

```
unsigned char pixel ;
pixel = image[115][28];
```

Ou encore, "balayer" tous les pixels d'une image s'effectue de la façon suivante (balayage ligne par ligne) :

```
for( y=0; y<HAUTEUR_IMAGE; y++ )
    for( x=0; x<LARGEUR_IMAGE; x++ )
        pixel = image[x][y];
```

2. Le programme

Le présent TP a pour objectif de réaliser un certain nombre de traitements de base sur des images. Un squelette de programme est fourni, afin de simplifier la mise en œuvre des méthodes.

Ce programme effectue les tâches suivantes :

1. Définition de variables et de constantes utiles ;
2. lecture d'une image source ;
3. écriture de l'image traitée ;
4. affichage de l'image traitée et de l'image source, à l'aide d'un programme externe.

L'étudiant devra donc insérer ses algorithmes de traitement, entre la lecture de l'image source (étape 2), et l'écriture de l'image traitée (étape 3).

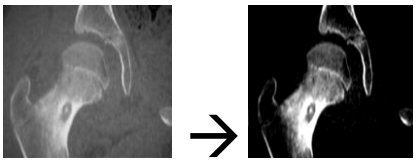
Les variables déclarées (cf. leur description ci-dessous) permettront d'orienter la réalisation des traitements.

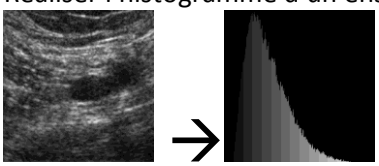
Type	Nom de la variable	Description
`unsigned char`	`image[LARGEUR][HAUTEUR]`	La matrice contenant l'image source
`unsigned char`	`image2[LARGEUR][HAUTEUR]` `imageTemp[LARGEUR][HAUTEUR]`	Deux matrices pour une seconde image, ou une image temporaire (traitements multiples)
`unsigned char`	`imageT[LARGEUR][HAUTEUR]`	La matrice de l'image après traitement
`unsigned char`	`imageC[LARGEUR][HAUTEUR]`	La matrice d'une éventuellement image couleur
`char`	`nomFichier[]`	Le nom du fichier image à lire, et qui sera utilisée pour tous les traitements. A modifier suivant les traitements, par exemple : `"images\\lena.raw"`
`char`	`nomFichier2[]`	Le nom du deuxième fichier image à lire. A modifier suivant les images utilisées, par exemple : `"images\\echoendo.raw"`
`char`	`nomFichierT[]`	Le nom du fichier image à écrire. Il s'agit du fichier qui contiendra le résultat du ou des traitements, par défaut `"images\\res.raw"`. Il n'est pas utile de modifier le contenu.
`char`	`nomFichierC[]`	Le nom du fichier de l'image couleur, par défaut `"images\\lena_color.raw"`
`int`	`x, y, i, j, n`	Variables d'itération et autres
`FILE*`	`fichier`	La variable fichier
`int`	`pixel`	Variable tampon pour stocker temporairement la valeur d'un pixel
`float`	`fpixel`	Variable tampon pour traitements nécessitant un type de flottant
`int`	`histogramme[256]` `histogramme_cum[256]`	Vecteur contenant l'histogramme Vecteur contenant l'histogramme cumulé
`unsigned char`	`median[MEDIAN*MEDIAN]`	Tableau pour le filtre médian
`char`	`str[256]`	Variable texte temporaire

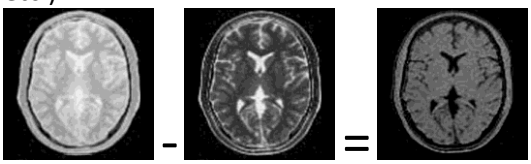
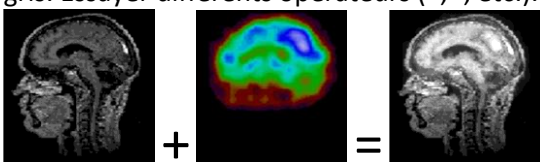
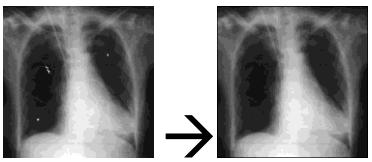
(les cases grisées indiquent les variables dont les valeurs seront à changer directement dans leur déclaration)

Traitements à implémenter

Voici les quelques traitements que nous vous demandons d'implémenter. Ils ne sont pas nombreux, aussi, prenez votre temps et essayez de bien comprendre leur intérêt ainsi que leur fonctionnement propre.

Méthode	Description	Image(s) source
Contraste	Effectuer un contraste sur une image radio d'un fémur, pour mettre en évidence des détails. 	`femur.raw`

Histogramme	Réaliser l'histogramme d'un ensemble d'images, afin d'en comparer les profils.
	 
		`femur.raw`
		`thrombose.raw`
		`radio.raw`
		`irm2.raw`

Fusion intra-modalité	Implémenter une fusion entre deux images. Ces deux images ont la même taille et sont en niveaux de gris. Essayer différents opérateurs (+, -, min, max, etc.). 	`irm1.raw` `irm2.raw`
Fusion inter-modalité	Implémenter une fusion entre deux images, cette fois, de modalités différentes. La première est en niveaux de gris, la seconde en couleur. L'image résultante étant en niveaux de gris, convertir le pixel de couleur en niveau de gris. Essayer différents opérateurs (+, -, etc.). 	`irm_bw.raw` `irm_c.raw`
Segmentation	Effectuer une segmentation d'une thrombose veineuse (image ultrason). Pour cela, réaliser un seuillage suivi d'un filtrage de type contour. Commencer par le seuillage, en testant différentes valeurs de seuil, jusqu'à obtenir celui qui vous paraît le plus pertinent (il est également possible d'effectuer une analyse d'histogramme). Comparer ensuite différents résultats provenant des différents filtrages cités plus haut. 	`thrombose.raw`
Ouverture	Effectuer une ouverture sur une image radio. 	`radio.raw`