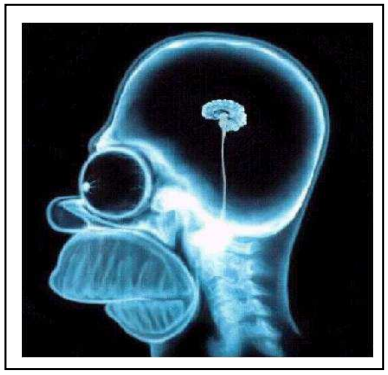
9 : LE TRAITEMENT D'IMAGES

- Représentation de l'information
- Algorithmique
- Langages et programmation
- Architectures matérielles



CONNAISSANCES ABORDEES

Numérisation : Modifier format, taille, contraste ou luminance d'images numériques

Algorithme simple : Modifier un algorithme existant pour obtenir un résultat différent

Algorithme simple : Concevoir et programmer un algorithme

Types de données : Choisir un type de donnée en fonction d'un problème à résoudre

Fonctions : Concevoir l'entête d'une fonction puis la fonction elle-même

PROBLEMATIQUE

Comment modifier une image pour améliorer son rendu et l'exploiter efficacement ?

CONDITIONS DE DEROULEMENT DE L'ACTIVITE

Phases de travail	***Objectifs***	***Activités***
A) Mise en situation	***Présenter** les différentes techniques d'imagerie médicale*	
B) Activités	***Etre capable** d'utiliser un programme existant et modifier une partie de son code* ***Réaliser** un algorithme de traitement en suivant un cahier des charges*	***Réalisation** d'algorithmes de traitement d'images*
C) Synthèse	***Rendre** compte de manière par écrit de son résultat*	***Synthèse** écrite avec copies d'écran*

LOGICIEL UTILISE :

- **Compilateur C DevC++ 4.9.9.2**

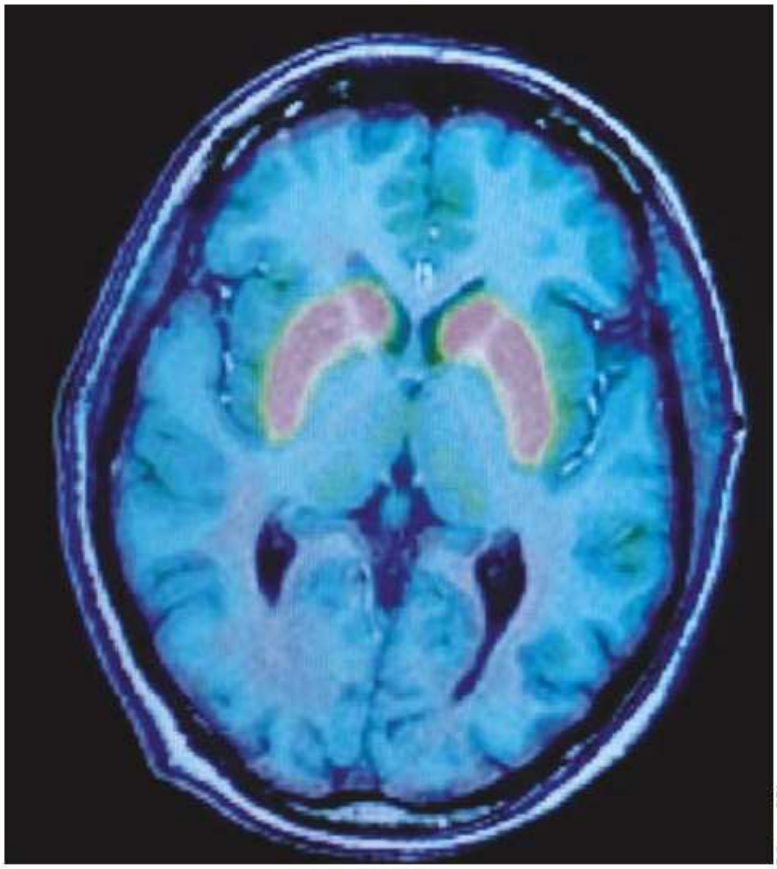
DUREE : 2 Séances

1. LES TECHNIQUES D'IMAGERIE MEDICALE

Les principales techniques d'imagerie médicale

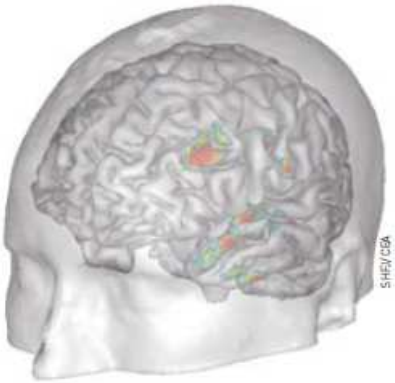
L'**imagerie médicale** est une méthode unique permettant de visualiser des processus biologiques au sein même des organismes vivants, de manière non invasive. Elle est essentielle à la compréhension de leur physiologie et de leurs pathologies afin de mieux les diagnostiquer, les pronostiquer et les soigner. L'imagerie constitue donc un outil d'investigation de choix de plusieurs champs de la médecine et de la biologie.
Initiée avec la radiographie par rayons X, l'imagerie médicale a bénéficié de la découverte de la **radioactivité** artificielle et des techniques de détection associées pour se développer. Par la suite, la découverte de la **résonance magnétique nucléaire** (RMN) puis des aimants **supraconducteurs** a permis des avancées technologiques significatives dans le domaine de l'**imagerie par résonance magnétique** (IRM).

Parmi les principales méthodes d'imagerie dynamique du cerveau humain, l'**électroencéphalographie** (EEG) permet de mesurer l'activité électrique du cerveau, provoquée par le courant généré dans les **neurones**, à l'aide d'**électrodes** placées sur le cuir chevelu (le scalp). Elle renseigne sur l'activité neurophysiologique du cerveau au cours du temps et en particulier du **cortex** cérébral, soit dans un but diagnostique en neurologie, soit dans la recherche en neurosciences **cognitives**.
La **magnétoencéphalographie** (MEG) enregistre les **champs magnétiques** induits par les courants générés par les neurones au moyen de capteurs positionnés à proximité de la tête. Employée dans un but clinique en neurologie, notamment pour le cas de l'épilepsie, ainsi que dans la recherche en neurosciences cognitives, cette technique autorise également l'étude de maladies développementales (dyslexie), psychiatriques (schizophrénie) et neurodégénératives (Parkinson, Alzheimer).
La **tomographie par émission de positons** (TEP) consiste à administrer par voie intraveineuse une **molécule** marquée avec un **isotope** radioactif afin de suivre, par détection externe, le fonctionnement normal ou pathologique d'un organe. Les **traceurs** radioactifs présentent les mêmes propriétés physico-chimiques que leurs homologues non radioactifs si ce n'est qu'ils possèdent la particularité d'émettre un rayonnement. Ils servent donc de balise pour suivre, à l'aide d'outils de détection appropriés, le cheminement d'une molécule préalablement marquée dans l'organisme. Les valeurs ainsi recueillies sont ensuite analysées et transformées à l'aide d'un modèle mathématique afin de permettre la reconstruction à l'écran d'une image représentant la position du radiotraceur dans l'organisme. La TEP est aujourd'hui largement utilisée pour des études physiologiques et physiopathologiques de la **cognition** et du comportement, ainsi que pour l'étude de différentes pathologies affectant le système nerveux central telles que l'épilepsie, l'ischémie cérébrale, les accidents vasculaires cérébraux et les maladies neurodégénératives (Parkinson, Huntington...).
L'**imagerie par résonance magnétique nucléaire** (IRM) est une méthode d'ima-



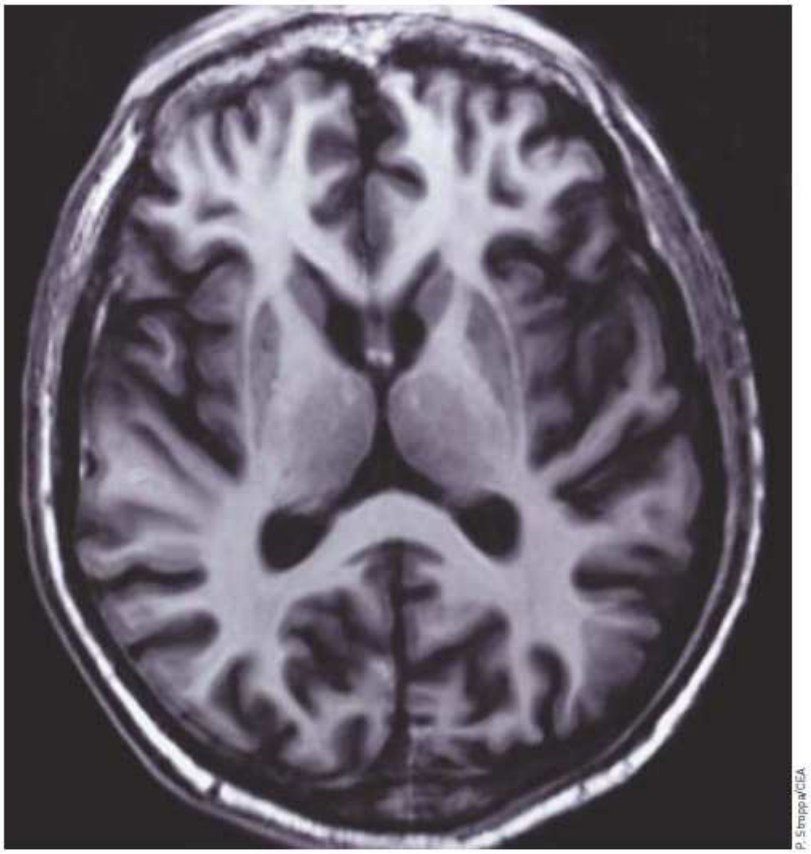
P. Stropp/CEA

Image en TEP. Les positons émis par les traceurs radioactifs préalablement injectés au patient sont détectés par la caméra TEP, ce qui permet, après analyse informatique, de reconstituer une image en 3D de l'organe étudié.



SHFJ/CEA

Dépression mélancolique. Fusion d'images en TEP mesurant l'activité énergétique régionale avec l'image en IRMa du cerveau d'un patient. Les zones hypoactivées sont détectées individuellement.



P. Stroppa/CEA

Image acquise avec le système IRM de 3 T du SHFJ situé à Orsay (Essonne). Cette technique permet une analyse très fine des lésions infectieuses ou inflammatoires, des anomalies des vaisseaux, ainsi que des tumeurs.

gerie fonctionnelle d'investigation *in vivo* non traumatique. Capable d'étudier des tissus dits mous, tels que le cerveau, la moelle épinière, les muscles, elle permet d'en connaître la structure anatomique, mais également d'en suivre le fonctionnement ou le métabolisme. Il s'agit dans le premier cas d'une IRM anatomique (IRMa), dans le deuxième d'une IRM fonctionnelle (IRMf) et dans le troisième de la spectroscopie IRM (SRM).

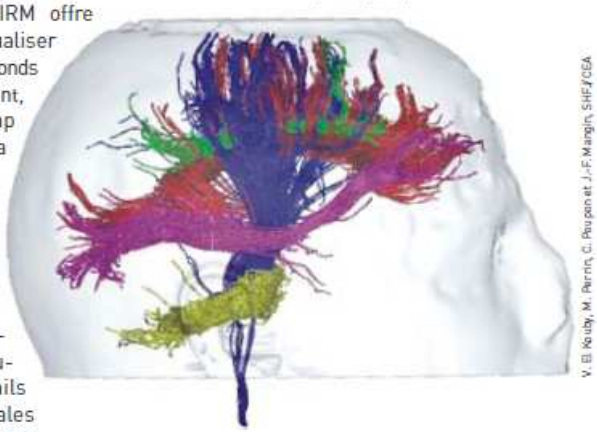
L'IRM utilise le phénomène de la RMN, technique de spectroscopie découverte en 1946 qui tire profit des propriétés magnétiques des noyaux atomiques. Certains noyaux, ceux d'hydrogène par exemple, sont dotés d'un petit moment magnétique ou spin. La RMN consiste à détecter les variations de l'aimantation des noyaux atomiques sous l'action d'un champ magnétique extrêmement puissant et d'une onde électromagnétique excitatrice. Lors de l'application d'une onde électromagnétique de fréquence adaptée, la *fréquence de résonance*, ces noyaux changent d'orientation puis émettent des signaux en retrouvant leur position d'origine. Avec les progrès de l'informatique et des champs magnétiques, la RMN est passée de la physique de la matière condensée à l'analyse chimique puis à la biologie structurale, et plus récemment à l'imagerie médicale.

L'IRM anatomique. L'IRM offre la possibilité de visualiser l'anatomie d'organes profonds et opaques. En observant, sous l'effet d'un champ magnétique intense, la résonance des noyaux d'hydrogène, présents en abondance dans l'eau et les graisses des tissus biologiques, cette technique permet en particulier de visualiser le cerveau en coupes montrant les détails des structures cérébrales (matière grise, matière blanche) avec une précision millimétrique. Cette imagerie "anatomique" est utilisée par les radiologues pour la détection et la localisation de lésions cérébrales.

L'IRM fonctionnelle. Plus récemment, grâce à la vitesse d'acquisition et de traitement de données, l'IRM est aussi devenue "fonctionnelle", révélant l'activité des différentes structures qui composent notre cerveau. Quand nous parlons, lisons, bougeons, pensons..., certaines aires de notre cerveau s'activent. Cette activation des neurones se traduit par une augmentation du débit sanguin local dans les régions cérébrales concernées. C'est cette augmentation locale et transitoire de débit sanguin, et non directement l'activité des neurones, qui peut être détectée par l'IRMf du fait de l'aimantation de l'hémoglobine contenue dans les globules rouges.

L'IRM de diffusion (IRMd). C'est un outil puissant pour mesurer, à l'échelle microscopique, les mouvements des molécules d'eau et établir ainsi l'architecture fine du tissu neuronal et de ses variations. Elle offre une mesure plus directe que les méthodes d'imagerie classiquement utilisées. Elle permet de sonder la structure des tissus à une échelle bien plus fine que la résolution des images IRM et se révèle plus rapide.

La spectroscopie par résonance magnétique nucléaire (SRM) complète cette palette de technologies en fournissant une méthode non invasive d'étude de la biochimie et du métabolisme du système nerveux central. Elle permet la quantification précise de plusieurs dizaines de molécules et est basée sur le même principe que l'IRM.



V. El Kouby, M. Perrin, C. Poupon et J.-F. Mangin, SHFJ/CEA

L'IRMd permet le diagnostic très précoce de certaines pathologies et la visualisation des faisceaux de fibres (matière blanche) qui relient les différentes régions cérébrales.

www.cea.fr/content/download/5416/35384/file/MemoC.pdf

2. PROGRAMME UTILISE

OBJECTIF

L'objectif est de lire une image numérique en format brut (.raw) afin de réaliser un certain nombre de traitements de base afin d'améliorer le rendu.

LE LANGAGE C EN TRAITEMENT D'IMAGES

Le langage C est le langage le plus communément utilisé dans le domaine du traitement d'images numériques.

La principale motivation du courant emploi de ce dernier, est sa rapidité d'exécution, car le traitement d'images requiert des ressources conséquentes, tant au niveau de la gestion de la mémoire, que de la charge processeur.

En effet, la majeure partie des traitements est basée sur le principe d'itérations (assez souvent des boucles imbriquées sont utilisées), dans la mesure où il faut traiter tous les pixels d'une image, qui est représentée par une matrice (ou un vecteur de vecteurs).

En outre, le langage C permet une grande maîtrise de la mémoire (gestion par pointeurs), et également, une optimisation des accès à cette mémoire (accès directs par pointeur). De plus, il est également possible de grandement optimiser les algorithmes.

CARACTERISTIQUES DE BASE DES IMAGES NUMERIQUES

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).

L'activité 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 28 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];
```

FONCTIONNEMENT DU PROGRAMME

Un squelette de programme est fourni (TPImage.cpp), afin de simplifier la mise en œuvre des méthodes.

Ce programme effectue les tâches suivantes :

1. *Définition de variables et de constantes utiles (cf. la liste ci-après),*
2. *Lecture d'une image source,*
3. *Ecriture de l'image traitée,*
4. *Affichage de l'image traitée et de l'image source, à l'aide d'un programme externe.*

Les algorithmes de traitement devront être insérés entre la lecture de l'image source (étape 2) et l'écriture de l'image traitée (étape 3).

(Toutes les variables ne seront pas utilisées dans cette activité)

Type	**Nom de la variable**	**Description**
unsigned char	image[LARGEUR][HAUTEUR]	La matrice contenant l'image source
unsigned char	image2[LARGEUR][HAUTEUR]	Une matrice 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\\resultat.raw". Il n'est pas utile de modifier cette valeur
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 lors de calculs
float	fpixel	Variable tampon pour traitements nécessitant un type de flottant (dans le cas de divisions par exemple)
int	histogramme[256]	Vecteur contenant l'histogramme
unsigned char	median[MEDIAN*MEDIAN]	Tableau pour le filtre médian
char	str[256]	Variable texte temporaire

EXEMPLES DE TRAITEMENT POSSIBLE :

Méthode	Description
Recopie	Recopier l'image source dans l'image traitée, pixel par pixel.
Traitements spatiaux	- Créer une copie miroir de l'image source (inversion sur l'axe X). - Créer une copie inversée de l'image source (inversion sur l'axe Y).
Négatif	Créer une copie négative de l'image source.
Luminosité	Créer une copie de l'image source dont on modifie la luminosité (augmenter, diminuer).
Seuillage	Créer une copie de l'image source sur laquelle on effectue un seuillage. Tester différentes valeurs (127, 64, 192, etc.).
Contraste	Créer une copie de l'image source dont on modifie le contraste (augmenter, diminuer).
Conversion en niveaux de gris	Créer une copie en niveaux de gris, d'une image couleur. L'image couleur est codée sur 3 octets, en RVB (Rouge, Vert, Bleu), et est chargée dans la matrice `imageC`. Tester également la décomposition en R, V, B, en ne copiant qu'un canal de couleur dans l'image traitée.
Filtrages moyenneurs	Effectuer un filtrage sur diverses images. Essayer différents paramétrages, dont : $\frac{1}{9}\begin{vmatrix}1&1&1\\1&1&1\\1&1&1\end{vmatrix}$ Moyennage 3*3 $\frac{1}{25}\begin{vmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{vmatrix}$ Moyennage 5*5 $\frac{1}{16}\begin{vmatrix}1&2&1\\2&4&2\\1&2&1\end{vmatrix}$ Gaussien
Filtrage médian	Effectuer un filtrage médian sur diverses images. Utiliser la variable " `median` " pour stocker le tableau temporaire de valeurs, que l'on trie avec la fonction " `qsort(median,MEDIAN*MEDIAN,sizeof(unsigned char), compare);` ". La fonction `compare` est fournie en début de fichier.
Filtrage contours	$\begin{vmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{vmatrix}$ Contours verticaux $\begin{vmatrix}-1&-1&-1\\0&0&0\\1&1&1\end{vmatrix}$ Contours horizontaux $\frac{1}{3}\begin{vmatrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{vmatrix}$ Laplacien
Gaussien, Laplacien	Enchaîner les traitements Gaussien et Laplacien
Contraste, Laplacien	Enchaîner les traitements de contraste et Laplacien. Les intervertir.
Superposition	- Superposer deux images, " `image` " et " `image2` ", dans " `imageT` ". - Les superposer en prenant en compte que les pixels de valeur " `220` " de " `image2` " sont transparents.
Histogramme	Créer l'histogramme de diverses images. L'histogramme sera stocké logiquement dans la variable " `histogramme` ", puis sauvegardée par l'intermédiaire de `imageT`, grâce à une recopie du contenu de l'une dans l'autre.

EXEMPLES DE RESULTAT :

NEGATIF

LUMINOSITE

CONTRASTE

FILTRE CONTOUR

3. ACTIVITE : LE TRAITEMENT D'IMAGES BIOMEDICALES

RECOPIE DE L'IMAGE :

Ouvrir le fichier TPImage.cpp, compiler et tester le programme.

Compléter le programme afin de recopier à l'identique (nommé resultat.raw) votre image de départ.

AFFICHAGE DE L'HISTOGRAMME :

Compléter le programme en ajoutant les lignes de code suivantes :

```
// Initialisation de l'histogramme à "0"
for
( i=0; i<256; i++ )
histogramme[i] = 0;
// Initialisation du rendu de l'histogramme à "0"
for
( y=0; y<HAUTEUR; y++ )
for
( x=0; x<LARGEUR; x++ )
imageT[x][y] = 0;
// Calcul de l'histogramme
int
max = 0;
for
( y=0; y<HAUTEUR; y++ )
{
for
( x=0; x<LARGEUR; x++ )
{
histogramme[image[x][y]]++;
if
( max<histogramme[image[x][y]] )
max = histogramme[image[x][y]];
}
}
// Rendu de l'histogramme
for
( x=0; x<LARGEUR; x++ )
{
for( y=HAUTEUR-1; y>HAUTEUR-((float)histogramme[x]/max*255); y--)
imageT[x][y] = x;
}
```

Compiler et tester le programme.

Améliorer ce programme pour permettre l'affichage sur la console (avant le lancement de l'histogramme) de la valeur de niveau de gris la plus présente dans l'image (ainsi que le nombre de pixels de cette valeur).

LUMINOSITE :

Pour augmenter ou diminuer la luminosité de l'image, il suffit d'augmenter (ou diminuer) la valeur de tous les pixels de l'image. Attention toutefois à ce que la valeur des pixels reste bien compris entre 0 et 255.

Compléter le programme afin d'augmenter la luminosité de l'image echoendo.raw. Essayer plusieurs valeurs (20,50,90 …) et constater le résultat.

NEGATIF :

Pour réaliser le négatif d'une image, il faut modifier la valeur de chaque pixel par effet miroir (0 -> 255, 240 -> 10 …).

Compléter le programme afin de réaliser le négatif de l'image lena.raw.

SEUILLAGE :

Le seuillage est une opération très simple qui consiste à affecter à un pixel une valeur "binaire" à partir d'un seuil : si le pixel courant est supérieur à ce seuil (par exemple 147), alors le pixel résultat vaudra 255, et sinon, il vaudra 0 (on obtient alors une image en noir et blanc, sans niveaux de gris).

Compléter le programme afin de réaliser le seuillage des images lena.raw et echoendo.raw pour plusieurs valeurs de seuils.

Comment serait-il possible d'automatiser la valeur optimale du seuil ?

CONTRASTE :

Modifier le contraste d'une image consiste à appliquer à chaque pixel un coefficient multiplicateur (par exemple 5). Un coefficient supérieur à "1" augmentera le contraste et inférieur diminuera le contraste. Nous touchons donc là à l'**histogramme** de l'image.

Ensuite, il faut bien comprendre que si l'on multiplie bêtement chaque pixel par ce coefficient, nous obtiendrons un histogramme qui pourra être très étendu et surtout, si nous multiplions par 5 par exemple, chaque pixel verra sa luminosité augmentée. Ce n'est pas ce que nous souhaitons obtenir, puisqu'au contraire, nous voulons que les pixels sombres le soient davantage et que les pixels lumineux le soient d'autant plus. Cette réflexion laisse entrevoir l'utilisation d'un **seuil** qui définira grosso modo ce qu'est un pixel sombre et un pixel clair (dans notre exemple, le seuil est de "128").

Il faudra donc ramener la moitié de l'histogramme des valeurs dites sombres sous "0" (valeurs négatives) et les valeurs dites lumineuses au-dessus de "0" (valeurs positives). Par conséquent, lorsque nous appliquerons par la suite le coefficient multiplicateur, les valeurs négatives le seront d'autant plus et les valeurs positives également. Il ne faut pas oublier d'ajouter la valeur du seuil à la fin de l'opération.

Dernière étape, il s'agit de reborner l'ensemble de l'histogramme sur [0..255].

Si vous souhaitez utiliser une valeur de contraste comprise entre 0 et 1, il faudra passer par des nombres flottants en utilisant la variable tampon fpixel et en pensant à convertir les différentes variables en (float) ou en (unsigned char).

Réaliser une modification de contraste sur les images cyto3.raw et echoendo.raw pour différentes valeurs de contraste (entiers).

FILTRE MOYENNEUR :

Il faut ici - pour le pixel courant [x][y] dans l'image destination - effectuer la moyenne des pixels adjacents, sur un masque de taille 3*3. La première étape est donc de récupérer les pixels du masque.

Pour ce faire, nous pouvons écrire les accès aux pixels du masque de la façon suivante :

(la case grisée représente le centre du masque)

[x-1][y-1]	[x][y-1]	[x+1][y-1]
[x-1][y]	[x][y]	[x+1][y]
[x-1][y+1]	[x][y+1]	[x+1][y+1]

A partir de ce schéma, il est facile de déduire la formule écriture dans le code exemple. La moyenne est simplement effectuée.

Pour effectuer un filtrage moyenneur 5*5, il suffit d'étendre la grille du masque, toujours centrée sur [x][y]. Pour filtrage gaussien, il faut en plus multiplier chaque valeur de pixel par le coefficient adéquat, indiqué préalablement dans l'énoncé.

Notons enfin que ces filtrages doivent tenir compte des effets de bord. En effet, pour un masque 3*3, il n'est pas possible de calculer la valeur moyennée pour le pixel [0][0] par exemple (le pixel [-1][-1] n'existe pas !). De fait, il faut commencer la boucle à 1 dans notre exemple (et à 2 si le masque est 5*5).

Réaliser sur l'image lena.raw un filtrage moyenneur avec un masque de taille 3*3.

FILTRE CONTOUR (LAPLACIEN) :

Le principe est le même que le filtre moyenneur précédent mais avec des coefficients suivants :

$$\frac{1}{3}\begin{vmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{vmatrix}$$

Réaliser un filtre contour sur les images lena.raw, echoendo.raw et echodoppler.raw.

4. POUR SE FAIRE PLAISIR ☺ …

GAUSSIEN + LAPLACIEN :

Il suffit en effet d'enchaîner les deux traitements, en utilisant une image intermédiaire, en l'occurrence image2 qui va servir pour stocker le résultat du traitement gaussien, avant de l'utiliser comme base pour le traitement laplacien.

CONTRASTE + LAPLACIEN :

Il suffit là encore d'enchaîner en utilisant l'image temporaire une opération de contraste suivie de l'application d'un filtre Laplacien.

SUPERPOSITION D'IMAGES :

SOLUTION 1 :

Réaliser pixel par pixel la moyenne des images. Constater et conclure.

SOLUTION 2 :

Une autre solution consiste sur l'image 2 à connaitre la valeur du gris autour de la mésange. Pour cela, il faudrait faire un histogramme (ici cette valeur vaut 220). Les pixels de l'image résultat correspondent soient aux pixels de l'image 1 si les pixels de l'image 2 sont grises (220), soient dans le cas contraire aux pixels de l'image 2.

Tester et conclure.

5. POUR ALLER PLUS LOIN : REDUIRE LE BRUIT D'UNE IMAGE

Réaliser un filtre moyenneur 3*3 puis 5*5 sur les images len_bruit.raw et len_bruitfort.raw. Conclure.

Une méthode pour réduire ce bruit est d'appliquer un filtrage median. Le calcul de l'image "médian" se compose de trois parties essentielles :

1. **Création de la liste des valeurs** : cette étape stocke dans un tableau l'ensemble des valeurs du masque médian. En l'occurrence, dans notre exemple, le masque étant de 3*3, nous stockons 9 valeurs dans le tableau défini pour (median). L'opération est assez longue d'écriture et difficilement lisible, mais il s'agit d'une simple généralisation de la récupération des valeurs du masque lorsqu'on le déplace sur l'image, en utilisant deux nouvelles boucles for.

2. **Tri du tableau** : le tableau fraîchement produit doit ensuite être trié. En effet, la valeur médiane sera finalement celle qui sera au centre du tableau trié dans l'ordre croissant (ou décroissant). La procédure de tri est simple, pour peu que l'on connaisse l'utilisation de la fonction C standard qsort. Cette dernière est déclarée de la façon suivante : qsort(tableau_a_trier, taille_du_tableau, taille_dun_element, fonction_de_comparaison).

Le tableau est celui calculé auparavant, sa taille est MEDIAN*MEDIAN (3*3 dans notre exemple), la taille d'un élément est celle d'un unsigned char et enfin, la fonction de comparaison est celle qui se trouve dans le code, en haut, à savoir compare.

3. **Affectation de la médiane** : la valeur médiane n'a plus qu'à être affectée, en sélectionnant celle qui se trouve en milieu de tableau.

6. SITOGRAPHIE

Bruit d'une image
http://www.01net.com/editorial/282533/comment-ca-marche-le-bruit-dimage/

Site de G. Brunet ENSTA BRETAGNE sur le traitement d'images :
http://public.enst-bretagne.fr/~brunet/

Tri rapide QSort() :
http://www.siteduzero.com/tutoriel-3-36691-le-tri-rapide-qsort.html

Imagerie médicale CEA :
www.cea.fr/content/download/5416/35384/file/MemoC.pdf