Université de Paris Sud

Polytech' APP5 **TI**

Filtrage, détection de contours et morphologie mathématique

1 Introduction

Le but de ce TP est de d'implanter en C différentes opérateurs de TI et de les tester sur des images réelles et de synthèse.

Les codes C fournis sont séparés en modules :
- `nrutil` : extension de la bibliothèque d'allocation mémoire de NRC. Permet en particulier le chargement et l'écriture d'image au format PGM
- `mutil` : *misc util* : routines diverses
- `gauss` : formule de Gauss en 1D et 2D
- `filterNRC` : operateurs de filtrage : filtrage gaussien, filtrage moyen, filtrage médian
- `histogramNR` : manipulation d'histogramme, principalement seuillage optimal d'Otsu
- `lutNR` : fonctions de seuillage à base de *LooK Up Table*
- `contourNR` : opérateurs de détection de contours
- `morphoNR` : opérateurs de morphologie mathématique
- `noise` et `random` : générateurs aléatoires
- `test` : fonction principale de test
- `my_path` : header définissant les chemins d'accès aux différents répertoires
- `test_filterNR` : fonctions de test des opérateurs de filtrage
- `test_contourNR` : fonctions de test des opérateurs de détection de contours
- `test_morphoNR` : fonctions de test des opérateurs morphologiques

Ce sont dans les fichiers `filterNRC`, `lutNRC` et morphoNR que vous devez remplir les fonctions vides. C'est dans le fichier `my_path` que vous devez spécifier les chemins effectifs sur votre machine. Enfin ce sont dans les fichiers `test_filterNR`, `test_contourNR` et `test_morphoNR` que vous devez ajouter de nouveaux tests. Pour cela, copiez le bloc de code et modifier le paramètre (en première ligne de chaque bloc) afin de réaliser un traitement supplémentaire. Le découpage est le suivant : toute fonction préfixée par `test_` contient l'appel à une routine de traitement et lui passe en argument les paths nécessaires ainsi que le nom de l'image utilisée (sans bruit) et avec bruit. Toute fonction C préfixée par `routine_` contient un traitement avec un jeu de paramètre donné (1 valeur de sigma pour le filtrage guassien, 1 taille de noyau pour le lissage moyen ou pour le lissage médian, ...).
De même, il y a un certains nombre de répertoires prédéfinis :
- `src` : répertoire contenant les images sources
- `noise` : répertoire contenant les images bruitées
- `denoise` : répertoire contenant les images débruitées
- `contour` : répertoire contenant les images de contours, en niveau de gris (après l'opérateur de détection de contours), et en binaire (après seuillage)
- `mrohpo` : répertoire contenant les images après traitement morphologique

Dans un premier temps, le but est de coder un certains nombre d'opérateurs et de les appliquer sur des images sans bruit ou faiblement bruitées. Dans un second temps, le but est de combiner des pré-traitements et des post-traitements afin d'améliorer les images de contours.

Le compte rendu, doit contenir des explications sur la démarche suivie ansi que votre analyse sur la façon dont vous avez géré le bruit. Vos explications seront illustrés par des exemples de résultats avec à partir d'image faiblement bruitées et d'autres fortement bruitées. La prise d'initiative dans la combinaison d'opérateur sera évaluée.
Votre compte-rendu sera une archive `zip` (exclusivement) à votre nom. Cette archive devra contenir

un répertoire à votre nom, contenant les différences codes C ainsi que le comptre rendu au format `pdf`.L'archive sera à envoyer par mail à l'adresse `lionel.lacassagneu-psud.fr`. Le sujet de ce mail sera `APP5_TI`. Le respect de ces consignes ainsi que la qualité des commentaires (dans le compte-rendu) et dans les codes sources sera prise en compte dans l'évaluation.
Pour visualiser les images au format PGM (binaire) : différents *viewers* sont disponibles comme Irfanview ou XnView.

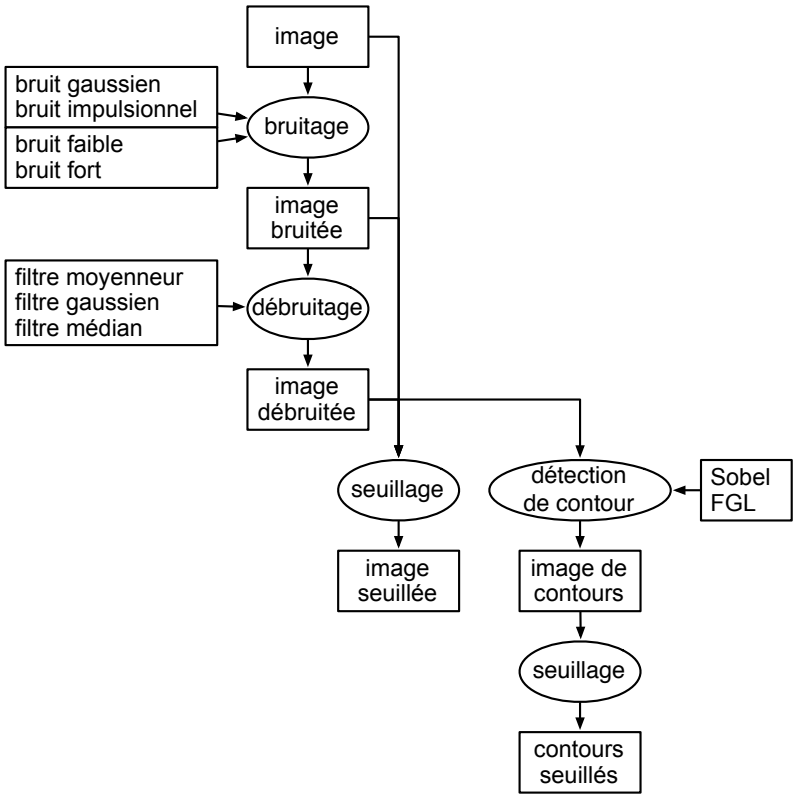

FIGURE 1 – étapes de traitement

2 Travail à faire

Le travail à faire reprend les différentes étapes de la figure 1.

1. ajouter les codes d'opérateurs manquant dans les fichiers `filterNRC`, `lutNRC` et morphoNR
2. ajouter des noms d'images à traiter ou à retraiter (image avec bruit gaussien ou avec bruit impulsionnel) dans les fonctions `test_`
3. ajouter des nouveaux appels à des fonctions de traitement (nouvelle valeur de paramètre) dans les fonctions `routine_`

2.1 Bruitage

Les fonctions de bruitage (bruit gaussien, bruit impulsionnel) sont fournies, tout comme la fonction pour générer des images contenant un carré dont le niveau de gris est séparé de celui du fond d'un certain écart.

1. Bruiter les images de synthèse avec du bruit gaussien et du bruit impulsionnel. Un exemple est fourni, en faire d'autres.
2. qu'observe-t-on ?

2.2 Débruitage

1. Coder la fonction effectuant un opérateur de lissage de type moyenneur (*average* en anglais) en s'inspirant du lissage gaussien fourni.
2. Coder la fonction effectuant un opérateur de lissage de type médian (basé sur un tri classique).
3. tester les filtres de débruitage sur du bruit additionnel ou du bruit impulsionnel

2.3 Détection de contours

1. Coder la fonction effectuant le calcul de la norme L1 du gradient de Sobel.
2. Coder la fonction effectuant le calcul de la norme L1 du gradient de FGL.
3. tester les deux opérateurs sur des images sans bruit, puis avec bruit.
4. seuiller les images (soit avec un seuil manuel, soit avec l'opérateur d'Otsu)

3 Morphologie Mathématique

1. Implanter les opérateurs suivants :
   (a) érosion et dilatation, pour des images binaires et en niveaux de gris,
   (b) ouverture et fermeture (qui réaliseront des appels aux fonctions précédentes),
   (c) filtres alternés séquentiels, basés sur un ensembles d'ouverture et sur un ensemble de fermetures
2. tester les différents opérateurs sur des images avec ou sans bruit impulsionnel.

4 Rappels

4.1 Average

$$A = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{1}$$

4.2 Sobel

$$G_x = \frac{1}{8}\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad G_y = \frac{1}{8}\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \qquad G = |G_x| + |G_y| \tag{2}$$

4.3 FGL

Le filtre 1D de FGL est composé d'un filtre causal et d'un filtre anticausal (Eq. 4).

$$y_+(n) = b_0 x(n) + a_1 y(n-1) + a_2 y(n-2) \tag{3}$$
$$y_-(n) = b_0 x(n) + a_1 y(n+1) + a_2 y(n+2) \tag{4}$$

Les coefficients (Eq. 5) seront calculés en virgule fixe (multiplication par 256). Le paramètre du filtre est α (Eq. 6).

$$b_0 = 256 \times (1-\gamma)^2 f_0, \quad a_1 = 256 \times 2\gamma, \quad a_2 = -256 \times \gamma^2 \tag{5}$$

avec

$$\gamma = e^{-\alpha}, \quad \alpha \in [0.5 : 0.9] \tag{6}$$

Le dérivateur de FGL est composée des gradients de Roberts (Eq. 7).

$$G_x = \frac{1}{4}\begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix} \qquad G_y = \frac{1}{4}\begin{pmatrix} -1 & -1 \\ 1 & 1 \end{pmatrix} \qquad G = |G_x| + |G_y| \tag{7}$$