TP n°1
Prise en main d'une librairie de traitement d'images

==========

L'objectif de ce premier TP est de manipuler et traiter des images à partir d'une libraire de traitement des images

Les TP se dérouleront sous LINUX avec un terminal, un éditeur de texte et un logiciel de tracé de courbes GNUPLOT. Il est fortement conseillé de créer un répertoire TP_image_1 et de tout sauvegarder dans le même répertoire.

1) Seuillage d'une image au format pgm

A partir des programmes téléchargés depuis : http://www2.lirmm.fr/~wpuech/enseignement/master_informatique/Analyse_Traitement_Image/TP/librairie/

a) Ouvrer ces fichiers avec un éditeur de texte et regarder leurs contenus.
b) Ces programmes manipulent des images au format pgm et ppm. Rechercher des informations sur ce type de format. Télécharger des images au format pgm depuis : http://www2.lirmm.fr/~wpuech/enseignement/master_informatique/Analyse_Traitement_Image/TP/images/
c) Compiler le programme principal et exécuter le.

   En fait, le programme téléchargé permet de seuiller une image en 2 parties :

   Si p(i,j) < Seuil
   alors p(i,j) = 0, (noir)
   Sinon p(i,j) = 255 (blanc)

d) Tester plusieurs valeurs de seuil.

2) Seuillage d'une image pgm avec plusieurs niveaux S1, S2, S3

A partir du programme utilisé dans la première partie, écrire un nouveau programme afin de seuiller une image :

a) en 3 parties
b) en 4 parties.

Pour trois parties :

Si p(i,j) < S1
alors p(i,j) = 0
Sinon Si p(i,j) < S2 alors p(i,j) = 128 (gris)
Sinon p(i,j) = 255

Dans le cas où S1 < p(i,j) < S2, proposer une autre valeur pour p(i,j).

Tester plusieurs valeurs de S1 et S2, puis S1, S2 et S3.

3) Profil d'une ligne ou d'une colonne d'une image pgm

Ecrire un programme profil permettant d'afficher à l'écran sur 2 colonnes les indices et les niveaux de gris d'une ligne ou d'une colonne d'une image. Cette programme aura comme arguments, le nom l'image, une information précisant s'il s'agit d'une ligne ou d'une colonne, et un indice indiquant le numéro de la ligne ou de la colonne.

Au lieu d'afficher les valeurs des pixels d'une ligne ou d'une colonne à l'écran, nous souhaitons maintenant les visualiser sous forme de courbes à l'aide du logiciel GNUPLOT.
Pour cela, il faut rediriger l'affichage de l'écran vers un fichier que nous appellerons `profil.dat` qui sera ensuite visualisé avec le logiciel Gnuplot.
Visualiser les profils d'une ligne et d'une colonne d'une image au format pgm indiquée par l'enseignant.
A l'aide du logiciel GNUPLOT, pour visualiser une courbe :

```
plot 'fich.dat' with lines
```

4) Histogramme d'une image pgm

A partir d'une image en niveau de gris au format pgm, écrire un programme permettant de sauver dans un fichier les données de l'histogramme de cette image. Le fichier contiendra 2 colonnes : indice et occurrence des niveaux de gris.
A l'aide du logiciel GNUPLOT, visualiser l'histogramme :

```
>plot 'histo.dat' with lines
```

5) Histogrammes des 3 composantes d'une image couleur (ppm)

A partir d'une image en couleur au format ppm, écrire un programme permettant de visualiser à l'écran, puis de sauver dans un fichier, les données des histogrammes des trois composantes couleur (Rouge, Vert, Bleu) de cette image.
Le fichier contiendra 4 colonnes : indice (de 0 à 255) et occurrence du rouge, puis du vert et enfin du bleu. A l'aide du logiciel GNUPLOT, visualiser sur un même graphique les trois histogrammes

6) Seuillage d'une image couleur (ppm)

A partir d'une image en couleur au format ppm, écrire un programme permettant de seuiller séparément chacune des trois composantes couleurs (Rouge, Vert, Bleu) de cette image.
Visualiser l'image de sortie pour différentes valeurs de seuils.

7) Seuillage automatique d'une image pgm

Reprendre la question 1) en proposant un ca lcul automatique de la valeur du seuil.

TP n°2
Opérations morphologiques sur des images

==========

L'objectif de ce TP est de continuer à manipuler et traiter une image à partir d'une libraire de traitement des images en langage C. Les TP se dérouleront sous LINUX avec un terminal, un éditeur de texte et un logiciel de tracé de courbes GNUPLOT.

1) Seuillage d'une image et érosion de l'image binaire

A partir des programmes `test_grey.cpp` et `image_ppm.h` téléchargés :

a) Prendre l'image au format pgm indiquée par l'enseignant, seuiller cette image en testant plusieurs valeurs de seuils. Modifier le programme `test_grey.cpp` afin que le fond de l'image (pixels < seuil) soit en blanc (255) et que les pixels des objets soient en noir (0). Comparer l'image seuillée avec au moins 3 valeurs différentes et en déduire le seuil le plus pertinent.

b) A partir de l'image seuillée la plus intéressante, écrire le programme `erosion.cpp`, qui va permettre de supprimer les points objets isolés. Compiler ce programme et l'exécuter en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant. Pour l'exécution, 3 arguments sont nécessaires, à savoir, le nom du programme, le nom de l'image d'entrée et le nom de l'image de sortie.

2) Seuillage d'une image et érosion de l'image binaire

a) A partir de l'image seuillée la plus intéressante obtenue à la question 1.a, écrire le programme `dilatation.cpp`, qui va permettre de boucher des petits trous isolés dans les objets de l'image. Compiler ce programme et l'exécuter en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant. Pour l'exécution, 3 arguments sont nécessaires, à savoir, le nom du programme, le nom de l'image d'entrée et le nom de l'image de sortie.

3) Fermeture et ouverture d'une image de l'image binaire

La fermeture d'une image binaire consiste à enchainer une dilatation et une érosion sur l'image binaire. Cela permet de boucher des trous dans les objets contenus dans l'image binaire.

L'ouverture d'une image binaire consiste à enchainer une érosion et une dilatation sur l'image binaire. Cela permet de supprimer des points parasites du fond de l'image binaire.

a) A partir de l'image seuillée la plus intéressante obtenue à la question 1.a, écrire le programme `fermeture.cpp`, qui va permettre de boucher des petits trous isolés dans les objets de l'image. Compiler ce programme et l'exécuter en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant. Pour l'exécution, 3 arguments sont nécessaires, à savoir, le nom du programme, le nom de l'image d'entrée et le nom de l'image de sortie.
b) A partir de l'image seuillée la plus intéressante obtenue à la question 1.a, écrire le programme `ouverture.cpp`, qui va permettre de supprimer des points parasites du fond de l'image binaire. Compiler ce programme et l'exécuter en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant. Pour l'exécution, 3 arguments sont nécessaires, à savoir, le nom du programme, le nom de l'image d'entrée et le nom de l'image de sortie.
c) Enchainer la fermeture et l'ouverture sur la même image, c-à-d que l'image de sortie du premier programme sera utilisée comme image d'entrée du second programme. Que constatez-vous ?
d) Afin d'avoir plus d'impact, nous vous proposons d'amplifier les effets fermeture et ouverture sur l'image binaire. L'idée est d'appliquer séquentiellement à l'image binaire, 3 dilatations, 6 érosions et enfin 3 dilatations.
    d)..a Dans une première approche, reprendre les programmes `erosion.cpp` et `dilatation.cpp` et les appliquer 6 fois dans l'ordre proposé.
    d)..b Ecrire un nouveau programme unique qui enchainera les 3 dilatations, les 6 érosions et enfin les 3 dilatations.
    d)..c Comparer les 2 images obtenues. Que constatez-vous ?

4) Segmentation d'une image

Le but de cette question est de développer une approche permettant de visualiser les contours d'une image. A partir de l'image seuillée la plus intéressante obtenue à la question 1.a et de l'image dilatée obtenue à la question 2, écrire une programme `difference.cpp` qui va nous permettre de visualiser les contours des objets contenus dans l'image :

Si les deux pixels (de l'image seuillée et de l'image dilatée) appartiennent au fond, alors le pixel correspondant de l'image de sortie appartiendra au fond (255).

Si les deux pixels (de l'image seuillée et de l'image dilatée) appartiennent à l'objet, alors le pixel correspondant de l'image de sortie appartiendra au fond (255).

Si le pixel de l'image dilatée appartient à l'objet et que le pixel de l'image seuillée appartient au fond, alors le pixel correspondant de l'image de sortie appartiendra au contour (0).

a) écrire le programme `difference.cpp`, compiler ce programme et l'exécuter en utilisant l'image binaire et l'image dilatées obtenues précédemment avec le seuil le plus intéressant. Pour l'exécution, 4 arguments sont nécessaires, à savoir, le nom du programme, les noms des deux images d'entrée et le nom de l'image de sortie.

5) Extension aux images en niveaux de gris, puis en couleur

Le but de cette question est d'étendre les questions précédentes (érosion, dilatation, fermeture et ouverture) dans un premier temps aux images en niveau de gris (c-a-d sans seuiller l'image), puis aux images couleur.

6) Question bonus : calcul de distance

A partir d'une image binaire écrire un programme afin d'effectuer des mesures de distances sur un objet : 1) calcul du volume englobant, 2) distance de l'échiquier, 3) distance de Manathan.

TP n°3
Spécification d'histogramme

==========

L'objectif de ce TP est d'effectuer des prétraitements sur une image à partir de méthodes de transformation d'histogramme.

==========

1) Expansion dynamique

A partir de l'image black.pgm, tracer l'histogramme, puis effectuer une expansion dynamique. Donner les valeurs de $\lceil$ et $\rfloor$.

Rendre : image black.pgm, histo black.pgm, valeurs de $\lceil$ et $\rfloor$, image black'.pgm et histo black'.pgm.

Appliquer le même algorithme sur les 3 composantes de l'image black.ppm

Rendre : valeurs de $\lceil$r, $\rfloor$r, $\lceil$g, $\rfloor$g, $\lceil$b, $\rfloor$b et image black'.ppm

2) Seuillage des extrema des trois histogrammes

A partir d'une image couleur couvrant tous les niveaux de gris, il n'est pas possible d'appliquer directement une expansion dynamique. Par contre il possible de couper les extrémités des 3 distributions et d'appliquer ensuite une expansion dynamique.
Prendre une image couleur quelconque, visualisation les histogrammes des composantes rouge, verte et bleue, et décider de des valeurs de seuils (Smin et Smax différentes pour chacune des composantes) afin de supprimer les valeurs extrêmes sur chacun des histogrammes (par exemple, tous les pixels ayant une valeur inférieure à Smin seront mis à Smin et tous les pixels ayant une valeur supérieure à Smax seront mis à Smax. Mais il existe d'autres solutions plus intéressantes)
Visualiser à nouveau les histogrammes.

Appliquer ensuite l'expansion dynamique comme dans la question 1.

Rendre : image et histo de l'image originale.pm, l'image et l'histo de l'image_seuillée.ppm, l'image et l'histo de l'image finale.

3) Egalisation d'histogramme

A partir d'une image en niveau de gris au format pgm, tracer l'histogramme, puis la densité de probabilité. En déduire la fonction de répartition F(a) de cette image (valeurs dans un tableau).

A partir de F(a), calculer T(a) pour chacun des niveaux de gris de l'image et l'appliquer à l'image.

Rendre : image_originale, histo image_originale, ddp, courbe de F(a), image_égalisée et histo image_égalisée.

4) Spécification d'histogramme

L'image lena.pgm sera considérée comme image de référence R.
Choisir une seconde image B et effectuer une spécification d'histogramme par rapport à R.

Rappel : effectuer l'égalisation d'histogramme de B à partir de sa propre fonction de répartition, puis effectuer une transformation inverse en utilisant la fonction de répartition inverse de R.

Rendre : image B, histo image B, image B égalisée, histo image B égalisée, image B spécifiée, histo image B spécificié (superposé avec histo lena en utilisant replot).

TP n°4
Floutage du fond d'une image

==========

L'objectif de ce TP est de flouter le fond d'une image contenant un objet ou un personnage en premier plan. Pour cela il faut, dans un premier temps segmenter le personnage du fond de l'image, pour, dans un second temps flouter uniquement le fond de l'image.

1) Création d'une image couleur au format ppm

A partir d'Internet, télécharger une image réelle (c-a-d pas une image de synthèse) contenant en premier plan un personnage (plutôt clair ou foncé) en un fond texturé (plutôt foncé si personnage clair, sinon plutôt clair). Transformer cette image au format ppm de taille 512x512 pixels.

Dans votre compte rendu :

- **Insérer l'image couleur en précisant l'adresse web de référence.**

2) Création d'une image en niveau de gris à partir d'une image couleur.

a) A partir de votre image couleur, créer un programme permettant d'avoir une image en niveaux de gris.
b) Avec l'aide de GNUPLOT, afficher l'histogramme de l'image en niveau de gris.

Dans votre compte rendu :

- **Ecrire la formule de transformation utilisée,**
- **Insérer l'image couleur en niveau de gris,**
- **Insérer l'histogramme.**

3) Seuillage de l'histogramme

A partir de l'histogramme obtenu à la question précédente :

a) Proposer un seuil afin de partitionner l'image en 2 classes (le fond et l'objet ou le personnage),
b) Seuiller l'image en niveau de gris avec la valeur proposer afin d'obtenir en sortie une image binaire (objet ou personnage en noir et fond en blanc).

Remarque : ce n'est pas grave si une partie du personnage reste en blanc et qu'une partie du fond soit en noir.

Dans votre compte rendu :

- **Ecrire l'algorithme permettant de seuiller l'image en niveau de gris,**

- **Proposer une valeur de seuil adaptée à votre image,**
- **Insérer l'image binaire obtenue.**

4) Floutage de l'image couleur

a) Ecrire un programme permettant de flouter l'image couleur dans sa globalité.

En fait sur chacune des composantes de l'image, chaque pixel de l'image va être remplacé par la valeur moyenne calculée avec ses voisins.
Dans le cas d'un voisinage 3*3, nous aurons :

p'(i,j) = 1/9 [p(i-1,j-1) + p(i-1,j) + p(i-1,j+1)
+ p(i,j-1) + p(i,j) + p(i,j+1)
+ p(i+1,j-1) + p(i+1,j) + p(i+1,j+1)].

Le filtre est donc de la forme : 1/9 * |1 1 1|
|1 1 1|
|1 1 1|

b) Flouter l'image couleur.

Remarque : à vous de juger le nombre de voisins nécessaires pour que le flou soit visible ou sinon appliquer plusieurs fois votre programme.

Dans votre compte rendu :

- **Ecrire l'algorithme permettant de flouter un pixel couleur,**
- **Insérer l'image floutée obtenue.**

5) Floutage du fond de l'image couleur

a) Ecrire un programme permettant de flouter uniquement le fond de l'image couleur. En entrée de votre programme il vous faut utiliser l'image couleur ainsi que l'image binaire obtenue à la question 3.

Dans votre compte rendu :

- **Ecrire l'algorithme permettant de flouter un pixel si et seulement il appartient au fond,**
- **Insérer l'image floutée partiellement.**

6) Erosion et dilatation

a) A partir de l'image binaire, écrire un programme combinant érosion et dilatation afin de supprimer des trous et des parasites.

Dans votre compte rendu :

- **Ecrire l'algorithme présentant cette combinaison,**
- **Insérer l'image binaire traitée.**

b) Reprendre la question 5 en utilisant l'image binaire traitée.

TP n°5
Segmentation d'une image par Split and Merge

==========

L'objectif de ce TP est de comprendre la segmentation d'une image avec une approche basée région : split and merge

1) Division d'une image en 4 régions

Télécharger une image réelle en couleur. Transformer cette image au format ppm de taille 512x512 pixels, puis en pgm.

A partir de l'image en niveau de gris, créer un programme qui divise l'image en 4 régions de taille égales. Pour chaque région calculer la valeur moyenne ainsi que la variance.

Afficher dans l'image de sortie la valeur moyenne de chaque région.

Dans votre compte rendu :

- **Insérer l'image d'entrée en niveau de gris,**
- **Insérer l'image de sortie contenant les 4 valeurs moyennes,**
- **Indiquer la variance de chaque région.**

2) Etape de division récursive

Au niveau de la première question, nous nous rendons compte que les 4 régions sont pas ou peu homogènes. Il faut diviser à nouveau chacune des régions en 4 nouvelles sous-régions et recalculer les valeurs moyennes ainsi que les variances.

A partir de l'image en niveau de gris, écrire une fonction récursive qui reçoit une région de l'image et une valeur de seuil, divise la région en 4 sous-régions, calcule la moyenne et la variance de chaque région et rappelle la fonction si la variance est supérieure à la valeur du seuil. Attention, la fonction doit s'arrêter si la taille de la région devient trop petite.
L'image finale contiendra pour chaque région la valeur moyenne de sa région.

Dans votre compte rendu :

- **Ecrire l'algorithme récursif proposé,**
- **Insérer plusieurs images divisées en faisant varier la valeur du seuil.**

3) Etape de fusion

Après l'étape de division nous pouvons nous rendre compte que certaines régions voisines ont des caractéristiques voisines (valeurs moyennes relativement proches). Il faut donc les fusionner. Pour cela, ajouter pendant la phase de division un graphe d'adjacence (RAG). A chaque région est associé un sommet du graphe, et des arêtes relient les sommets correspondants à deux régions qui sont voisines.

Pour l'étape de fusion, pour chaque sommet de RAG, il faut alors chercher s'il existe un sommet voisin dans le RAG et de valeur suffisamment proche, et si c'est le cas, on les fusionne. Pour savoir si deux régions ont des valeurs proches, nous calculerons la différence des valeurs moyennes et nous comparerons cette différence à un seuil.

Dans votre compte rendu :

- **Ecrire l'algorithme de fusion,**
- **Proposer une valeur de seuil adaptée à votre image,**
- **Insérer l'image fusionnée obtenue.**

4) Image couleur

Si vous avez le temps, et encore assez d'énergie, étendre votre méthode aux images couleurs.

TP n°2
Filtre inverse vidéo et floutage d'images

==========

L'objectif de ce TP est de continuer à manipuler et traiter une image à partir d'une libraire de traitement des images en langage C. Les TP se dérouleront sous LINUX avec un terminal, un éditeur de texte et un logiciel de tracé de courbes GNUPLOT. Il est fortement conseillé de créer un répertoire TP_image_2 et de tout sauvegarder dans le même répertoire.
http://www.lirmm.fr/~wpuech/enseignement/DUT_SRC/1A/traitement_images/librairie

1) Inverse vidéo

A partir des programmes `test_grey.cpp` et `image_ppm.h` téléchargés, il est demandé de créer un nouveau programme `inverse.cpp` qui va inverser les niveaux de gris d'une image. En fait le noir (0) doit devenir blanc (255), et le blanc doit devenir noir.

a) Ecrire le programme `inverse.cpp`, le compiler et l'exécuter en utilisant l'image indiquée par l'enseignant. Pour l'exécution, 3 arguments sont nécessaires, à savoir, le nom du programme, le nom de l'image d'entrée et le nom de l'image de sortie.
b) Visualiser les profils d'une ligne des 2 images (entrée et sortie). Comparer les 2 profils. Que constatez-vous ?

2) Filtre flou1

Créer un nouveau programme `filtre_flou1.cpp` qui va remplacer, dans l'image de sortie, la valeur d'un pixel par la valeur moyenne de ce pixel avec ses 4 voisins :
`p_out(i,j)= (p(i,j)+p(i-1,j)+p(i+1,j)+p(i,j-1)+p(i,j+1))/5.`
Il est fortement recommander de faire attention aux bords de l'image. En effet, la première et la dernière colonnes, ainsi que la première et la dernière lignes ne peuvent pas être traitées, il faudra conserver les valeurs initiales.

3) Filtre flou2

Créer un nouveau programme `filtre_flou2.cpp` qui va remplacer, dans l'image de sortie, la valeur d'un pixel par la valeur moyenne de ce pixel avec ses 8 voisins.

4) Analyse des résultats

a) Visualiser les profils d'une ligne d'une image obtenue avant et après application des filtres flous 1 et 2.
b) Appliquer 3 fois le filtre flou 2 sur la même image. Pour cela, avec le même programme, l'image de sortie devient l'image d'entrée pour l'itération suivante. Comparer les différents profils de lignes obtenus.

TP n°3
Erosion et dilatation d'une image

==========

L'objectif de ce TP est de continuer à manipuler et traiter une image à partir d'une libraire de traitement des images en langage C. Les TP se dérouleront sous LINUX avec un terminal, un éditeur de texte et un logiciel de tracé de courbes GNUPLOT. Il est fortement conseillé de créer un répertoire TP_image_3 et de tout sauvegarder dans le même répertoire.
http://www.lirmm.fr/~wpuech/enseignement/DUT_SRC/1A/traitement_images/librairie

1) Seuillage d'une image et érosion de l'image binaire

A partir des programmes `test_grey.cpp` et `image_ppm.h` téléchargés :

c) Prendre l'image au format pgm indiquée par l'enseignant, seuiller cette image en testant plusieurs valeurs de seuils. Modifier le programme `test_grey.cpp` afin que le fond de l'image (pixels < seuil) soit en blanc (255) et que les pixels des objets soient en noir (0). Comparer l'image seuillée avec au moins 3 valeurs différentes et en déduire le seuil le plus pertinent.
d) A partir de l'image seuillée la plus intéressante, écrire le programme `erosion.cpp`, qui va permettre de supprimer les points objets isolés. Compiler ce programme et l'exécuter en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant. Pour l'exécution, 3 arguments sont nécessaires, à savoir, le nom du programme, le nom de l'image d'entrée et le nom de l'image de sortie.

2) Seuillage d'une image et érosion de l'image binaire

b) A partir de l'image seuillée la plus intéressante obtenue à la question 1.a, écrire le programme `dilatation.cpp`, qui va permettre de boucher des petits trous isolés dans les objets de l'image. Compiler ce programme et l'exécuter en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant. Pour l'exécution, 3 arguments sont nécessaires, à savoir, le nom du programme, le nom de l'image d'entrée et le nom de l'image de sortie.

3) Fermeture et ouverture d'une image de l'image binaire

La fermeture d'une image binaire consiste à enchainer une dilatation et une érosion sur l'image binaire. Cela permet de boucher des trous dans les objets contenus dans l'image binaire.
L'ouverture d'une image binaire consiste à enchainer une érosion et une dilatation sur l'image binaire. Cela permet de supprimer des points parasites du fond de l'image binaire.

e) A partir de l'image seuillée la plus intéressante obtenue à la question 1.a, écrire le programme `fermeture.cpp`, qui va permettre de boucher des petits trous isolés dans les objets de l'image. Compiler ce programme et l'exécuter en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant. Pour l'exécution, 3 arguments sont nécessaires, à savoir, le nom du programme, le nom de l'image d'entrée et le nom de l'image de sortie.
f) A partir de l'image seuillée la plus intéressante obtenue à la question 1.a, écrire le programme `ouverture.cpp`, qui va permettre de supprimer des points parasites du fond de l'image binaire. Compiler ce programme et l'exécuter en utilisant l'image binaire obtenue précédemment avec le seuil le plus intéressant. Pour l'exécution, 3 arguments sont nécessaires, à savoir, le nom du programme, le nom de l'image d'entrée et le nom de l'image de sortie.
g) Enchainer la fermeture et l'ouverture sur la même image, c-à-d que l'image de sortie du premier programme sera utilisée comme image d'entrée du second programme. Que constatez-vous ?
h) Afin d'avoir plus d'impact, nous vous proposons d'amplifier les effets fermeture et ouverture sur l'image binaire. L'idée est d'appliquer séquentiellement à l'image binaire, 3 dilatations, 6 érosions et enfin 3 dilatations.
    d)..d Dans une première approche, reprendre les programmes `erosion.cpp` et `dilatation.cpp` et les appliquer 6 fois dans l'ordre proposé.
    d)..e Ecrire un nouveau programme unique qui enchainera les 3 dilatations, les 6 érosions et enfin les 3 dilatations.
    d)..f Comparer les 2 images obtenues. Que constatez-vous ?

4) Segmentation d'une image

Le but de cette question est de développer une approche permettant de visualiser les contours d'une image. A partir de l'image seuillée la plus intéressante obtenue à la question 1.a et de l'image dilatée obtenue à la question 2, écrire une programme `difference.cpp` qui va nous permettre de visualiser les contours des objets contenus dans l'image :

Si les deux pixels (de l'image seuillée et de l'image dilatée) appartiennent au fond, alors le pixel correspondant de l'image de sortie appartiendra au fond (255).

Si les deux pixels (de l'image seuillée et de l'image dilatée) appartiennent à l'objet, alors le pixel correspondant de l'image de sortie appartiendra au fond (255).

Si le pixel de l'image dilatée appartient à l'objet et que le pixel de l'image seuillée appartient au fond, alors le pixel correspondant de l'image de sortie appartiendra au contour (0).

b) écrire le programme `difference.cpp`, compiler ce programme et l'exécuter en utilisant l'image binaire et l'image dilatées obtenues précédemment avec le seuil le plus intéressant. Pour l'exécution, 4 arguments sont nécessaires, à savoir, le nom du programme, les noms des deux images d'entrée et le nom de l'image de sortie.

TD en IMAGE
SRC2

Réaliser d'un diaporama par groupe de 2 (ou 3) sur un thème précis du traitement des images parmi ceux vus en TP de première année :

- segmentation (par seuillage),
- le filtrage (floutage, inverse vidéo),
- les érosions et dilatations

Vous aurez à présenter lors de la dernière séance de cours (ou de TD) votre diaporama par groupe pour une durée de 5' (donc 5 diapos)

En 2009-2010

TP1 SRC2 : utilisation de la librairie de traitement d'images en C++
TP2 SRC2 : recherche sur Internet segmentation basée région et basée contour
TP3 SRC2 : calcul d'un histogramme et affichage avec gnuplot
TP4 SRC2 : tracé d'un profil de ligne ou de colonne d'une image

En 2010-2011

TP n°1
Reprise en main de la librairie de traitement des images
Application aux images couleur

TP n°2
Affichage d'un profil de ligne ou de colonne d'une image couleur

TP n°3
Expansion dynamique de l'histogramme d'une image couleur

==========

L'objectif de ce TP est d'effectuer des prétraitements sur une image couleur à partir d'une méthode de transformation d'histogramme.

==========

1) Création d'une image couleur au format ppm

A partir d'Internet, télécharger une image réelle (c-a-d pas une image de synthèse) et la convertir au format ppm de taille 512x512.

2) Histogrammes des 3 composantes d'une image couleur

A partir de l'image en couleur générée au format ppm, écrire un programme permettant de visualiser à l'écran, puis de sauver dans un fichier, les données des histogrammes des trois composantes couleur (Rouge, Vert, Bleu) de cette image.
Le fichier contiendra 4 colonnes : indice (de 0 à 255) et occurrence du rouge, puis du vert et enfin du bleu. A l'aide du logiciel GNUPLOT, visualiser sur un même graphique les trois histogrammes

3) Seuillage des extrema des trois histogrammes

A partir de la visualisation des histogrammes des composantes rouge, verte et bleue, décider de valeurs de seuils (S_{min} et S_{max} différentes pour chacune des composantes) afin de supprimer les valeurs extrêmes sur chacun des histogrammes.
En fait, tous les pixels ayant une valeur inférieure à S_{min} seront mis à S_{min} et tous les pixels ayant une valeur supérieure à S_{max} seront mis à S_{max}.
Visualiser à nouveau les histogrammes.

4) Expansion dynamique

A partir de l'image couleur traitée précédemment, effectuer une expansion dynamique (comme vue en cours). Donner les valeurs de $\neg\lambda$ et $\llcorner\chi$.

TP n°5
Tracé d'un profil de ligne ou de colonne d'une image
Création d'une première image de contours

==========

L'objectif de ce TP est d'observer les variations sur une ligne ou une colonne d'une image en visualisant des profils de lignes ou de colonnes d'images, puis de créer une image de contours basée sur l'analyse des profils de lignes ou de colonnes.

1) Création d'une image couleur au format ppm

A partir d'Internet, télécharger une image réelle (c-a-d pas une image de synthèse) et la convertir au format ppm de taille 512x512.

2) Profil d'une ligne ou d'une colonne d'une image couleur

Ecrire un programme profil permettant d'afficher à l'écran sur 4 colonnes les indices et les trois composantes couleurs (Rouge, Vert, Bleu) de tous les pixels d'une ligne ou d'une colonne d'une image. Ce programme aura comme arguments, le nom l'image, une information précisant s'il s'agit d'une ligne ou d'une colonne, et un indice indiquant le numéro de la ligne ou de la colonne.

Au lieu d'afficher les valeurs des pixels d'une ligne ou d'une colonne à l'écran, nous souhaitons maintenant les visualiser sous forme de courbes à l'aide du logiciel GNUPLOT.

Pour cela, il faut rediriger l'affichage de l'écran vers un fichier que nous appellerons `profil.dat` qui sera ensuite visualisé avec le logiciel Gnuplot.

Visualiser les profils d'une ligne et d'une colonne d'une image au format ppm indiquée créé dans la partie 1 du TP.

3) Création d'une image de sortie enregistrant les variations importantes suivantes les lignes ou les colonnes d'une image en niveau de gris.

a) A partir de votre image couleur, créer un programme permettant d'avoir une image en niveaux de gris.

b) Créer une image de différence suivant les lignes. Il s'agit en fait d'enregistrer dans l'image de sortie la valeur absolue de la différence entre un pixel et le pixel précédent sur la même ligne (mettre 0 pour les pixels de la première colonne).

c) Créer une image de différence suivant les colonnes. Il s'agit en fait d'enregistrer dans l'image de sortie la valeur absolue de la différence entre un pixel et le pixel précédent sur la même colonne (mettre 0 pour les pixels de la première ligne).

4) Création d'une image de sortie enregistrant les variations importantes suivantes les lignes ou les colonnes

A partir des observations effectuées sur les profils de lignes ou de colonnes (partie 2), développer un nouveau programme qui à partir d'une image d'entrée et d'un paramètre de `saut` générera une image de sortie **binaire** enregistrant :

- en noir (valeur 0) les pixels qui auront une variation supérieure à la valeur du `saut`.
- en blanc (valeur 255) les pixels qui auront une variation inférieure à la valeur du `saut`.

a) Générer une image pour la lecture suivant les lignes,

b) puis une image pour la lecture suivant les colonnes.

c) Proposer une combinaison des deux images binaires obtenues.

TP n°6
Floutage d'un fond d'une image

==========

L'objectif de ce TP est de flouter le fond d'une image contenant un personnage en premier plan. Pour cela il faut, dans un premier temps segmenter le personnage du fond de l'image, pour, dans un second temps flouter uniquement le fond de l'image.

1) Création d'une image couleur au format ppm

A partir d'Internet, télécharger une image réelle (c-a-d pas une image de synthèse) contenant en premier plan un personnage (plutôt clair ou foncé) en un fond texturé (plutôt foncé si personnage clair, sinon plutôt clair). Transformer cette image au format ppm de taille 300x300 pixel.

Dans votre compte rendu :

- **Insérer l'image couleur en précisant l'adresse web de référence.**

2) Création d'une image en niveau de gris à partir d'une image couleur.

a) A partir de votre image couleur, créer un programme permettant d'avoir une image en niveaux de gris.
b) Avec l'aide de GNUPLOT, afficher l'histogramme de l'image en niveau de gris.

Dans votre compte rendu :

- **Ecrire la formule de transformation utilisée,**
- **Insérer l'image couleur en niveau de gris,**
- **Insérer l'histogramme.**

3) Seuillage de l'histogramme

A partir de l'histogramme obtenu à la question précédente :

c) Proposer un seuil afin de partitionner l'image en 2 classes (le fond et le personnage),
d) Seuiller l'image en niveau de gris avec la valeur proposer afin d'obtenir en sortie une image binaire (personnage en noir et fond en blanc).

Remarque : ce n'est pas grave si une partie du personnage reste en blanc et qu'une partie du fond soit en noir.

Dans votre compte rendu :

- **Ecrire l'algorithme permettant de seuiller l'image en niveau de gris,**
- **Proposer une valeur de seuil adaptée à votre image,**

- **Insérer l'image binaire obtenue.**

4) Floutage de l'image couleur

c) Ecrire un programme permettant de flouter l'image couleur dans sa globalité.
d) Flouter l'image couleur.

Remarque : à vous de juger le nombre de voisins nécessaires pour que le flou soit visible ou sinon appliquer plusieurs fois votre programme.

Dans votre compte rendu :

- **Ecrire l'algorithme permettant de flouter un pixel couleur,**
- **Insérer l'image floutée obtenue.**

5) Floutage du fond de l'image couleur

b) Ecrire un programme permettant de flouter uniquement le fond de l'image couleur. En entrée de votre programme il vous faut utiliser l'image couleur ainsi que l'image binaire obtenue à la question 3.

Dans votre compte rendu :

- **Ecrire l'algorithme permettant de flouter un pixel si et seulement il appartient au fond,**
- **Insérer l'image floutée partiellement.**

6) Erosion et dilatation

c) A partir de l'image binaire, écrire un programme combinant érosion et dilatation afin de supprimer des trous et des parasites.

Dans votre compte rendu :

- **Ecrire l'algorithme présentant cette combinaison,**
- **Insérer l'image binaire traitée.**

d) Reprendre la question 5 en utilisant l'image binaire traitée.

TP n°1

Seuillage d'images

==========

L'objectif de ce TP est de partitionner une image à partir de méthodes de seuillage d'images.

==========

1) Histogramme d'une image

A partir d'une image en niveau de gris au format pgm, écrire une méthode permettant de sauver dans un fichier les données de l'histogramme de cette image. Le fichier contiendra 2 colonnes : indice et occurrence des niveaux de gris.
A l'aide du logiciel GNUPLOT, visualiser l'histogramme :

```
>plot 'histo.dat' with lines
```

2) Seuillage d'images

Choisir une image bimodale en niveau de gris au format pgm. Ecrire une méthode afin de seuiller une image en 2 parties : Si p(i) < S alors p(i) = 0, sinon p(i) = 255.
La valeur de seuil S est choisie manuellement par rapport à l'histogramme. Tester plusieurs valeurs de S.

3) Seuillage d'images avec plusieurs niveaux S1, S2 (,S3)

Ecrire une méthode afin de seuiller une image en 3 parties ou 4 parties.
Pour trois parties :

Si p(i) < S1 alors p(i) = 0
Sinon Si p(i) < S2 alors p(i) = 128
Sinon p(i) = 255

Dans le cas où S1 < p(i) < S2, proposer une autre valeur pour p(i).
Tester plusieurs valeurs de S1 et S2, puis S1, S2 et S3.

4) Seuillage automatique d'une image

L'objectif de cette partie est de trouver automatiquement la valeur du seuil S.

a) Ecrire une méthode afin d'effectuer un suréchantillonnage de l'image. Passer en argument d'entrée la nouvelle valeur n du nombre de niveaux de gris :

p(i) = p(i) - p(i)%largeur + valeur_centrale.
Par exemple pour n=10, largeur = 256/n = 26 et valeur_centrale = 256/(2n) = 13.
Tester cette méthode avec des valeur comprises entre 10 et 50.

b) A partir de cette nouvelle image suréchantillonnée, trouver une méthode automatique afin de trouver une valeur pour S (choisir une image bimodale au départ).

TP n°2
Spécification d'histogramme

==========

L'objectif de ce TP est d'effectuer des prétraitements sur une image à partir de méthodes de transformation d'histogramme.

==========

1) Expansion dynamique

A partir de l'image black.pgm, tracer l'histogramme, puis effectuer une expansion dynamique. Donner les valeurs de $\neg_{\lambda}$ et $\llcorner_{\chi}$.

Rendre : image black.pgm, histo black.pgm, valeurs de $\neg_{\lambda}$ et $\llcorner_{\chi}$, image black'.pgm et histo black'.pgm.

2) Egalisation d'histogramme

A partir d'une image en niveau de gris au format pgm, tracer l'histogramme, puis la densité de probabilité. En déduire la fonction de répartition F(a) de cette image (valeurs dans un tableau). A partir de F(a), calculer T(a) pour chacun des niveaux de gris de l'image et l'appliquer à l'image.

Rendre : image_originale, histo image_originale, ddp, courbe de F(a), image_égalisée et histo image_égalisée.

3) Spécification d'histogramme

L'image lena.pgm sera considérée comme image de référence R.
Choisir une seconde image B et effectuer une spécification d'histogramme par rapport à R.
Rappel : effectuer l'égalisation d'histogramme de B à partir de sa propre fonction de répartition, puis effectuer une transformation inverse en utilisant la fonction de répartition inverse de R.

Rendre : image B, histo image B, image B égalisée, histo image B égalisée, image B spécifiée, histo image B spécificié (superposé avec histo lena en utilisant replot).

TP n°3
Filtrage d'images bruitées

==========

Le filtrage d'images a pour objectif de supprimer le bruit contenu dans une image. L'objectif de ce TP est d'observer les effets d'ajout de bruit sur des profils de lignes d'images, puis de filtrer ces images.

==========

1) Profil d'une ligne ou d'une colonne d'une image

Ecrire une méthode `profil()` permettant de sauver sur 2 colonnes dans un fichier les indices et les ndg d'une ligne ou d'une colonne d'une image. Cette méthode aura comme arguments, le nom de l'objet image ou de la matrice de pixels, un booléen précisant s'il s'agit d'une ligne ou d'une colonne, et un indice indiquant le numéro de la ligne ou de la colonne.
Le profil (`profil.dat`) sera ensuite visualisé avec le logiciel Gnuplot.
Visualiser les profils d'une ligne et d'une colonne des images lena.pgm et synthese.pgm.

2) Bruit blanc additif

Un bruit blanc additif est un bruit qui se rajoute au signal avec une probabilité uniforme.
Ecrire une méthode `bruit_blanc()` qui rajoute un bruit blanc dans une image. Cette méthode aura au moins comme argument l'amplitude du bruit ajouté. Par exemple, si `ampli = 5`, alors le bruit ajouté sera un nombre aléatoire compris entre –5 et +5.
Appliquer cette méthode à l'image synthese.pgm avec différentes valeurs pour l'amplitude. Pour chacune de ces images visualiser l'histogramme et les profils d'une ligne et d'une colonne.

3) Filtre moyenneur

Chaque pixel de l'image va être remplacé par la valeur moyenne calculée avec ses voisins.
Dans le cas d'un voisinage 3*3, nous aurons :

p'(i,j) = 1/9 [p(i-1,j-1) + (p(i-1,j) + (p(i-1,j+1)
+ p(i,j-1) + (p(i,j) + (p(i,j+1)
+ p(i+1,j-1) + (p(i+1,j) + (p(i+1,j+1)].

Le filtre est donc de la forme : 1/9 * |1 1 1|
|1 1 1|
|1 1 1|

Ecrire une méthode `filtre_moyenneur()` permettant de filtrer une image. Cette méthode aura au moins comme argument la taille du filtre (`t`, nombre impair >= 3).
Dans le cas d'un filtre 3*3, quelle est la conséquence sur le bord de l'image ? Proposer une solution.
Appliquer ce filtre avec les tailles 3*3, 5*5 et 9*9 sur l'image lena.pgm puis sur l'image synthese.pgm bruitée (avec 2 amplitudes pour le bruit).
Visualiser pour chacun de ces traitements, l'histogramme et les profils d'une ligne et d'une colonne. Conclure.

TP n°4
Détection des contours d'une image avec utilisation du gradient (1er ordre)

==========

L'objectif de ce TP est de traiter une image par la méthode des gradients afin d'en extraire les contours.

==========

1) Création de la carte de gradient d'une image

Ecrire une méthode `norme_gradient()` qui en chaque point d'une image calcule les gradients horizontal et vertical, puis retourne la norme du gradient.
Créer alors une image de la norme du gradient. Visualiser le profil d'une ligne de l'image de la norme du gradient.

2) Extraction des maximums locaux par seuillage.

Extraire de l'image de la norme du gradient les valeurs maximales par seuillage. En déduire l'image des contours. Tester plusieurs valeurs pour le seuil.

3) Seuillage par hystérésis des maximums locaux.

Seuiller d'abord avec un seuil haut SH et conserver ensuite toutes les chaînes ayant au moins un point dont la norme du gradient est supérieur au seuil bas SB. Tester plusieurs valeurs pour SH et SB.

4) Prétraitement par filtrage.

Recommencer les trois étapes précédentes en appliquant en amont sur l'image un filtre moyenneur.

TP n°5
Détection des contours d'une image avec le Laplacien (2ème ordre)

==========

L'objectif de ce TP est de traiter une image en utilisant la dérivée seconde afin d'en extraire les contours. Les traitements seront appliqués sur les images lena.pgm et synthese.pgm.

==========

1) Filtrage d'une image

Filtrer l'image lena.pgm en utilisant un filtre moyenneur. Comparer le profil d'une ligne avant et après filtrage.

2) Calcul du laplacien

Pour chaque pixel p(i,j) (0<i<m-1 et 0<j<n-1) de l'image, calculer le laplacien en utilisant le masque vu en cours. Créer une image en ajoutant la valeur +128 pour chaque pixel.

3) Recherche des passages par zéro

Un passage par zéro correspond à un changement de signe entre deus pixels voisins. Effectuer la recherche des passages par zéro dans la direction du gradient afin d'obtenir une image des contours. Ai=p(i+1,j)-p(i,j), Aj=p(i,j+1)-p(i,j), G=max(|Ai|,|Aj|), D=arctan(|Aj|/|Ai|).

4) Recherche des passages par zéro et seuillage par hystérésis

Créer l'image des passages par zéro affectés de la norme du gradient. Effectuer un seuillage par hystérésis afin d'éliminer les passages par zéro non significatifs.