Travaux pratiques - IMA201

Introduction

---

1 Préliminaires

1.1 Avant de commencer le TP

Il faut executer les commandes

```
export PATH=/cal/softs/anaconda/anaconda3/bin:$PATH
source /tsi/tp2/TP/Init/debut_tp_python.sh
spyder programmation/tp_initiation.py &
```

La première ligne vous permet d'utiliser python en version 3 distribué sous forme anaconda. Vous pouvez l'ajouter dans vos fichiers de configuration afin d'utiliser toujours cette version de python.

La seconde ligne crée un répertoire TPIMAINIT dans lequel se trouvera un lien symbolique vers le répertoire images. Vous serez automatiquement mis dans ce répertoire. Les images se trouvent dans le (lien symbolique vers le) répertoire images (dans lequel vous ne pouvez pas créer ou sauvegarder des images, créez-les dans votre répertoire de travail). Vous pouvez aussi bien sûr travailler sur les images de votre choix. La troisième ligne lance l'éditeur spyder sur le fichier fourni. Si vous utilisez votre propre ordinateur, vous pouvez vous contenter de spyder (ou tout autre environnement de travail python).

Vous trouverez dans le répertoire programmation un fichier tp_initiation.py qui contient quelques commandes pour commencer sous python. Il faut d'emblée exectuer les SECTIONS 1 et 2 afin d'inclure les packages nécessaires au TP. La SECTION 2 définit des fonctions utiles pour ce TP. Ensuite la SECTION 3 contient les commandes correspondant aux différentes questions de ce TP. Elle est à exécuter ligne par ligne et surtout vous devez comprendre ce que font ces lignes.

Vous pouvez utiliser autre chose que spyder, assurez-vous seulement que vous êtes dans le bon répertoire lorsque vous voulez accéder aux images. Les commandes python `pwd` et `cd` sont utiles pour cela.

2 Visualisation et utilisation de gimp

Le but de cette partie est de se familiariser avec le logiciel gimp.

gimp est un logiciel de traitement et d'édition d'images assez complet. Nous l'utilisons ici que pour visualiser les images et faire des opérations simples.

Par exemple pour connaître le niveau de gris d'un point (ou sa couleur) il faut sélectionner l'outil "pipette" puis aller dans le menu "Couleurs -> Courbes" (ce qui affichera un graphique). En cliquant sur un point de l'image le niveau de gris du point s'affiche dans le graphique.

2.1 Zooms

Ouvrez une image de votre choix dans gimp (`viewimage`). Zoomez-la d'un facteur 2 ou plus (menu déroulant "%" sous l'image) pour la visualiser en plus grand. Que fait gimp pour afficher l'image en plus grand?

Ouvrir l'image **lena.tif** et réduisez sa taille par le menu "Image -> Echelle..." d'un facteur deux. Attention: Dans le dialogue "Echelle..." sélectionner "Aucune" comme méthode d'interpolation. Zoomez-là pour l'afficher en deux fois plus grand.

Ouvrez l'image **lena_petit.tif** et zoomez-la d'un facteur 2 également.

Comparez le résultat par rapport au zoom de la petite image que vous avez produite. Quelle hypothèse pouvez-vous faire sur la génération de **lena_petit.tif**?

2.2 Espace couleurs

Ouvrir l'image **fleur.tif**(commande `viewimage_color`). Ouvrir le dialogue "Couleurs->Teinte-Saturation". Essayez de transformer les fleurs jaunes en fleurs bleues avec le bouton Hue. Comprenez-vous pourquoi les deux positions extrêmes de ce boutons font, en fait, la même transformation?

A quoi correspond la saturation (essayez-100% et +100%)?

3 Niveaux de gris, histogrammes et statistiques

Le but de cette partie est de mettre en oeuvre les notions radiométriques vues en cours : histogrammes, changements de contraste, égalisation et prescription d'histogramme.

3.1 Histogramme

Pour afficher l'histogramme d'une image on utilise la fonction `plt.hist(im.reshape((-1,)),bins=N)`, N étant le nombre de cases, voir l'exemple donné dans **tp_initiation.py**. Il faut préalablement convertir l'image en flottant `im=np.float32(im)`. Commencez par visualiser quelques histogrammes d'images de votre choix. Pour une image couleur, vous pouvez visualiser les histogrammes de l'un des canaux couleur. Pour extraire le canal rouge d'une image couleur *imcol* on peut procéder ainsi `rouge=imcol[:,:,1]`. Vous pouvez aussi visualiser l'histogramme de l'image obtenue en moyennant les canaux couleur `im=imcol.mean(axis=2)`.

Visualisez ensuite l'histogramme cumulé (noté H en cours) :

```
(histo,bins)=np.histogram(im.reshape((-1,)),np.arange(0,256))
histo=histo/histo.sum()
histocum=histo.cumsum()
```

Nous étudions maintenant l'effet de dégradations simples des images sur leur histogramme. A l'aide de la fonction `noise`, ajoutez un bruit gaussien à une image et étudiez l'effet produit sur son histogramme.

En considérant les niveaux de gris d'une image comme la réalisation d'une variable aléatoire dont la loi est l'histogramme de l'image, interprétez le résultat.

3.2 Changement de contraste

L'outils "Courbes" de Gimp (dans le menu "Couleurs"), permet de prescrire "à la main" un changement de contraste à appliquer à l'image (ainsi que de voir en gris clair l'histogramme de l'image). L'aspect global de l'image est-il modifié par l'application de fonctions croissantes ? Que se passe-t-il si l'on applique une transformation non-croissante des niveaux de gris?

En sélectionnant "luminosité-contraste" dans le menu Couleurs, vous pouvez voir l'effet de deux changements de contraste très simples :

- **luminosité** l'ajout d'une constante
- **contraste** la multiplication par une constante après centrage sur la valeur centrale de l'histogramme (i.e. $(im - 128) * k + 128$)

3.3 Egalisation d'histogramme

Comme vu en cours, l'égalisation d'histogramme consiste à utiliser comme changement de contraste l'hitogramme cumulé de l'image traitée. Visualisez son effet (par exemple sur l'image *sombre.jpg*) en reprenant l'histogramme cumulé *histocum* calculé précédemment :

```
imequal = histocum[np.uint8(im)]
```

Qu'observez-vous sur *imequal*, sur son histogramme et sur son histogramme cumulé.

3.4 Prescription d'histogramme

Comme vous l'avez vu en cours, une manière simple de donner à une image u l'histogramme d'une image v (en supposant que ces deux images ont mêmes dimensions) est d'utiliser les commandes suivantes :

```
ind=np.unravel_index(np.argsort(u, axis=None), u.shape)
```

```
unew=np.zeros(u.shape,u.dtype)
```

```
unew[ind]=np.sort(v,axis=None)
```

Les images **vue1.tif** et **vue2.tif** sont deux prises de vue d'une même scène avec la même ouverture et des temps d'expositions différents. Visualisez la valeur absolue de la différence des images, qu'observe-t-on. Même question après avoir donné à l'une des images l'histogramme de l'autre. A-t-on un moyen plus simple d'obtenir le même résultat (donner le même histogramme aux deux images), dans le cas particulier de ces deux images?

En vous inspirant des deux lignes de code ci-dessus, donnez un code simple permettant d'égaliser l'histogramme d'une image (le rendre aussi proche que possible d'une fonction constante).

3.5 Dithering

Le dithering (ou détramage) est une technique utilisée à l'origine pour obtenir un rendu en niveaux de gris lorsque l'appareil d'impression ne disposait que de deux couleurs (noir et blanc). A l'aide de la fonction `quantize`, observez l'effet d'une quantification de plus en plus brutale sur une image.

Pour comprendre la méthode du dithering, commencez par seuiller une image (avec `seuil`) et visualisez le résultat. Appliquez le même seuillage à une version bruitée de l'image originale et visualisez. Que constatez vous?

En considérant un pixel de niveau x dans l'image initiale, donnez la probabilité pour que ce pixel soit blanc après ajout de bruit et seuillage. Pourquoi l'image détramée ressemble-t-elle plus à l'image de départ que l'image simplement seuillée?

3.6 Différences de niveaux de gris voisins

On veut maintenant étudier la statistique de la différence de niveau de gris entre deux pixels adjacents. Grâce à la commande `gradx` on peut obtenir une image de la différence entre pixels adjacents. Visualisez l'histogramme d'une telle image. Visualisez le logarithme de l'histogramme.

La distribution des différences vous semble-t-elle obéir a une loi gaussienne ? Pourquoi ? Quelle aurait été la forme de l'histogramme si l'on avait considéré la différences entre pixels plus éloignés? (On ne demande pas de faire l'expérience).

4 Spectre des images et transformation de Fourier

4.1 Visualisation de spectres

Grâce à la fonction `view_spectre` on peut visualiser le spectre d'une image. Visualisez les spectres de différentes images en utilisant les options 1 et 2 (`help(view_spectre)`).

Que constatez-vous? Qu'en déduisez-vous par rapport au spectre d'une image? (Pour la suite on visualisera toujours avec l'option 2)

Comment influe l'option hamming sur le spectre de l'image? (multiplication de l'image originale par une fonction très lisse qui s'annule aux bords de l'image)

Visualisez le spectre de l'image synthétique **rayures.tif**. Que constatez-vous? Peut-on retrouver les caractéristiques des rayures de l'image à partir de son spectre? Expliquez la différence entre la visualisation avec et sans l'option hamming? Quel effet a le sous-échantillonnage sur le spectre (on peut utiliser une image synthétique et une image naturelle (Par exemple **carte_nb.tif**)).

4.2 Ringing

A l'aide de la fonction `filterlow`, appliquez un filtre passe bas parfait à une image. Visualisez l'image résultante, ainsi que son spectre. Que constatez-vous?

Mêmes questions en utilisant la commande `filtergauss`.

Visualisez les deux masques (sous le répertoire images) **masque_bas_centre.tif** (pour le filtrage passe-bas parfait) et **masque_gauss_centre.tif** (pour le filtrage gaussien). Quelle différence constatez-vous, en particulier quelle conséquence a la discontinuité de la transformée de Fourier sur la vitesse de décroissance du filtre spatial correspondant?